Universidad
Carlos III de Madrid
www.uc3m.es

TRABAJO FIN DE GRADO

Diseño de un sistema de control de relés de bajo coste para el testeo de equipos

Grado en Ingeniería Electrónica Industrial y Automática

Autor: Manuel Arias Ruiz

Tutor: Álvaro Castro González

Febrero 2014

Agradecimientos

A todas aquellas personas que han hecho posible este proyecto, que representa la culminación de la carrera de ingeniería, especialmente a mis compañeros de trabajo Octavio Campos, Antonio Oliva, Antonio López y Aldara Jodra, por su apoyo y dedicación.

A mi tutor Álvaro Castro, por la asignación del diseño y su constante ayuda en la elaboración del trabajo.

Por último, a mi familia y amigos, por hacer de este trayecto un camino liviano repleto de momentos inolvidables.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1 Introducción

En la actualidad se diseñan y fabrican equipos electrónicos complejos que requieren ser verificados, tanto en la fase de fabricación como en la fase de mantenimiento. Para poder llevar a cabo esta tarea de forma eficiente se diseñan y fabrican equipos de test que permiten la supervisión de estos equipos electrónicos de manera automática. La instrumentación electrónica desafía constantes variaciones y se ha transformado en una herramienta esencial para ingenieros, técnicos y científicos que requieren de sistemas electrónicos de medida y estimulación de gran precisión.

Por ejemplo encontramos equipos de test que incorporan fuentes de alimentación, tanto de continua como alterna, generadores de ondas, de pulsos, equipos de medida como multímetros, contadores, osciloscopios, etc.

La complejidad, cada vez mayor, de los diversos instrumentos y su veloz reorientación hacia el uso de tecnologías en común, propulsan a los sistemas de pruebas a lograr una mayor flexibilidad; sin embargo, el importe de los instrumentos exige un aumento de la longevidad de los dispositivos. La arquitectura modular, tanto *hardware* como *software*, proporciona el camino hacia unos dipositivos más económicos y adaptables a las necesidades. Los instrumentos modulares consisten en dispositivos que comparten elementos en común, *software* abierto a determinar por el consumidor y *bus*es de alta velocidad. Así, se satisfacen los requisitos de los sistemas de pruebas automatizadas *ATE* (*Automatic Test Equipment*) del presente y futuro.

Los instrumentos modulares comparten componentes como el chasis, la fuente de alimentación y la controladora, mientras que los instrumentos tradicionales e independientes duplican estos elementos en cada dispositivo, disparando el precio, el tamaño, el peso y disminuyendo la fiabilidad. Los componentes compartidos como procesadores y *software* mejoran el rendimiento con respecto al mismo uso en dispositivos tradicionales. Gracias a esta distribución, el tamaño se reduce considerablemente, por lo que el mismo espacio usado por instrumentos tradicionales no se usaría con tal provecho.

La arquitectura de la instrumentación modular convierte a los dispositivos en instrumentos dinámicos e intercambiables, más fáciles de modificar vía *software* para cambiar su función dependiendo de las necesidades de las pruebas. Además otorgan la ejecución a alta velocidad al explotar el rendimiento de los ordenadores industriales y las nuevas técnicas de temporización y sincronización.

En el presente, existen una gran variedad de instrumentos modulares que cumplen funciones muy diversas. Por ejemplo [48]:

- Los multímetros digitales (*DMMs*), están diseñados para funcionar en pruebas automatizadas, en *bus*es desde USB hasta *bus*es de estándares abiertos modernos. Los

multímetros digitales realizan medidas de resistencia, corriente, voltaje, capacitancia, inductancia e incluso temperatura de modo muy preciso.

- Los osciloscopios/digitalizadores, disponen de una mutabilidad inigualable en el manejo de la frecuencia y el tiempo, gracias a su arquitectura y una funcionalidad versátil definida por un *software* fácilmente modificable. Con un digitalizador es posible ejecutar pruebas propias de los osciloscopios y además usar el instrumento como un analizador de espectro, receptor ultrasónico y registrador de transitorios. También es factible sincronizar varios digitalizadores entre sí o con otros instrumentos de exactitud en muy poco tiempo, para aplicaciones de señal mixta o alto conteo de canales.
- Los dispositivos de E/S digital, brindan características muy evolucionadas para efectuar interfaces y ensayos de señal mixta y de sistemas digitales. Con instrumentos de E/S digitales basados en ordenadores personales, es viable la construcción sistemas de pruebas poderosos y flexibles para cumplir los objetivos marcados de la aplicación, desde estudios personalizados de comunicaciones hasta ensayos funcionales de fin de línea. Los dispositivos digitales disponen de características muy desarrolladas que permiten realizar pruebas como rendimiento dedicado en interfaces y validación interna de datos.
- Los instrumentos de RF, (Analizadores de redes verticales, generadores *multi-tonos*, *termocuplas*, analizadores de redes escalares, diodos detectores, bolómetros, etc.) abordan un amplio rango de pruebas necesarias como *WLAN, GPS, WiMAX. MIMO, ZigBee y RFID.* Permiten analizar las propiedades de amplitud y fase de las señales de radiofrecuencia.
- Los generadores de señal, son dispositivos capaces de crear señales de forma de onda arbitraria, en patrones periódicos o no periódicos, tanto analógicas como digitales, desde señales simples sinusoidales y de reloj hasta formas de onda complejas de comunicación moduladas por *I/Q*. Se suelen emplear en el diseño, ensayo y reparación de instrumentos electrónicos. Aunque tradicionalmente solían ser dispositivos muy estáticos difícilmente configurables, en la actualidad permiten la conexión y manipulación desde un PC, por lo que pueden ser gobernados mediante un *software* dedicado exclusivamente a la aplicación que realice, incrementando enormemente la flexibilidad.

Para que estos equipos de test puedan funcionar de manera automática es necesario llevar las señales y tomar las medidas de numerosos puntos. Por lo que se necesita un sistema de *switching* modular y adaptado a las características del sistema de test. Además, aumenta de manera considerable la reutilización del equipo.

Por ejemplo para señales de potencia como fuentes de alimentación es necesario usar relés de potencia, capaces de conmutar señales de altas corrientes y tensiones; para señales de frecuencias medias es preciso utilizar relés coaxiales que conservan las características de la señal sin añadir reflexiones y/o distorsiones; si seguimos aumentando en frecuencia y llegamos a la *RF* tenemos que usar otro tipo de relés específicos. Tampoco hay que olvidarse de los relés de propósito general para señales de baja frecuencia y baja potencia.

La industria actualmente proporciona distintos tipos de arquitectura de sistemas de *switching* basados en las arquitecturas de instrumentación que existen en el mercado. Por citar algunas:

- Sistemas de bajo coste, basados en arquitecturas de *PC, PCI, PCI- Express*, control por puerto serie o *USB*…
- Sistemas de coste medio basados en control por *Ethernet* o modulares como *PXI*.
- Sistemas de alta gama como *VXI*, usadas en ámbito militar.

Sin embargo, estos sistemas requieren de adecuaciones en cuanto a conectores y controladores, que incrementan mucho su precio. Por lo que es ineludible estudiar sistemas que sean capaces de reemplazarlos, conservando únicamente las características funcionales y evitando así la inflación del coste por parte de estos elementos.

De esta manera más del 90% del coste del sistema lo representa la funcionalidad, a diferencia de otras arquitecturas. Podemos ver esto con un ejemplo: un sistema de *switching* basado en *PXI. PXI* presume de ser una arquitectura de bajo coste; sin embargo, si queremos montar en nuestro equipo de test un sistema de *switching* basado en *PXI*, deberemos adquirir junto con los módulos de *switching* un chasis *PXI* y una controladora. Es probable que el coste de estos dos elementos aumente el precio de manera significativa, sin aportar una mejora en la funcionalidad. Si conseguimos la misma funcionalidad eliminando el coste de ambos elementos mejoramos la competitividad del producto al tener un coste significativamente menor.

Todo sistema de *switching* consta de dos partes fundamentales:

- Los elementos de conmutación o relés, controlados por señales discretas.
- La plataforma de control, incluyendo la arquitectura de las tarjetas o módulos.

Así, el diseño mostrará una clara diferenciación entre estas partes, pues simplifica mucho el planteamiento.

1.2 Motivaciones

La elección del tema del presente proyecto responde a varias circunstancias:

- La facilidad de acceso a la información.
- La dedicación al proyecto en el entorno de trabajo.
- El interés del tema.

Los *bus*es de estándar abierto *VXI, LXI* y *PXI* cuentan con una gran difusión y una enorme funcionalidad; sin embargo, condiciona al ingeniero a la utilización de la tecnología propia de estos *bus*es. Aunque esta tecnología es adecuada para el diseño modular, incrementa mucho los costes al comprometerse a adquirir componentes dedicados:

- *Carrier*, un armazón capaz de adaptarse a la arquitectura del instrumento, proveer una circuitería (contenida en un adaptador de interfaz de aplicaciones, que interactúa entre los módulos y la interfaz principal del sistema) dedicada a la alimentación y la

comunicación con el módulo o tarjeta, *software* adecuado a la aplicación que realiza y en ocasiones, un sistema de calibración de la circuitería.

- Chasis o *mainframe*, una plataforma que permite insertar los *carriers* y colaborar con su control.

La utilización de *GPIB* también supone un acondicionamiento a las características de este *bus* estandarizado, que también implica un alto coste (especialmente en la tarjeta *GPIB* y el *carrier* del instrumento).

Así, al diseñar un sistema modular sin emplear estas tecnologías se propicia un ahorro significativo.

1.3 Objetivos

El objetivo del proyecto es el diseño de un sistema de *switching* modular, de bajo coste, pero con prestaciones medias-altas. Es imprescindible que sea un sistema modular configurable y expandible capaz de dar solución a equipos de test de gama baja, media y alta.

Para conseguir que el sistema sea de bajo coste sin afectar a sus prestaciones el objetivo es abaratar el subsistema de control, utilizando para ello una tecnología muy barata y de uso universal como es *Ethernet*.

El diseño está enfocado a ser empleado en sistemas de prueba para dispositivos aeroespaciales militares, por lo que el sistema debe ser capaz de reemplazar en cierto grado el estándar utilizado para estos fines (*VXI*) y adaptarse en pequeña medida a este estándar de manera que también pueda ser compatible, es decir, debe mantener unas características mecánicas y unos conectores similares mientras que la comunicación debe ser transformada completamente a *Ethernet*.

El entorno de la implementación de estos equipos ya cuenta con plena funcionalidad y adaptación a un *hardware* determinado, por lo que el propósito del proyecto, además de proporcionar una alternativa más económica a los sistemas de prueba de aviónica, yace en la integración del sistema creado en la posición que ocupan los dispositivos *VXI* comerciales.

El proyecto se centra de esta manera en el diseño de un sistema de control modular de relés basado en *Ethernet* que consiste en:

- Tres módulos de relés de propósito general, denominados por su semejanza con algunas tarjetas de relés comerciales como:
    - *IND2002A*
    - *IND5002*
    - *IND5003*
- Un *driver Plug & Play*
- Una interfaz de usuario para *Windows*.

Si bien pueden parecer metas disociadas y fáciles de alcanzar, el solapamiento e integración de cada uno de los objetivos en el sistema implican un cometido más laborioso, que se encuentra determinado por una serie de requerimientos que condicionan su proyección.

Los requisitos del sistema se clasifican en tres categorías:

- Requisitos del *hardware*. El sistema debe mostrar compatibilidad con el formato de tamaño *VXI* más usado, el tamaño C (ver anexo A.1.2.1). Sin embargo, también se *bus*ca la versatilidad, por lo que es imperativo elaborar módulos de tarjetas de relés más pequeños, de manera que puedan situarse varios en el espacio de una ranura de *VXI* tamaño C. Cada módulo a proyectar posee unas características especiales que le hacen único y lo determinan a operar en un ámbito u otro.
- Requisitos del *software*. Los requisitos del *software* describen la manera en que el sistema se comunica con los elementos *hardware* que se desean gobernar, los relés. Esta transferencia de información se realiza a través de varias capas de *software*, por lo que los requerimientos se aplican a cada una de estas capas:
    - Interfaz gráfica de usuario *Windows*:
        - Sencilla, de manera que el usuario sea capaz de utilizar la interfaz sin tener plenos conocimientos sobre la manipulación de dispositivos electrónicos e informática.
        - Adaptable, de forma que la *GUI* se acomode a cualquier ordenador que cuente con un sistema operativo *Windows XP o Windows 7,* y *software VISA*.
        - Rápida, de modo que la interacción con la *GUI* sea veloz, y la transición entre pantallas y acciones ágil y dinámica.
        - Flexible, con la capacidad de incorporar módulos de relés de distinta configuración, añadir nuevas funcionalidades y nuevos menús.
        - Detallada técnicamente, de manera que las posiciones de los relés deben mostrar su correspondencia hacia los pines de la tarjeta.
        - Visible, de modo que todos los elementos de la interfaz se distingan claramente y no induzcan al error.
    - *Driver Plug & Play*:
        - Compatible, el controlador de instrumentos debe otorgar compatibilidad con las arquitecturas de *software* del estándar *VXI*.
        - Integrable, el *driver* debe presentar una arquitectura que facilite su inclusión y la llamada de sus funciones en programas de test complejos.
    - *Firmware*:
        - Amoldable, el *firmware* debe incorporar pleno funcionamiento con los módulos desarrollados en el proyecto; sin embargo, también debe permitir una fácil acomodación a futuros módulos con otras configuraciones.
        - Eficaz, la primera capa de *software* es el *firmware*, por lo que debe aprovechar todas las capacidades de la tarjeta de relés, ya que de otra forma se limitarían las competencias de los módulos con un cuello de botella.

- Requisitos comunes. Ciertas especificaciones tienen un carácter pragmático y pueden ser resueltas mediante el *hardware* o a través del *software*, o en ambos, por lo que constituyen características comunes. Los requisitos funcionales definen la funcionalidad o los servicios que el sistema debe otorgar:
    - En este proyecto es imperativo el uso de *Ethernet* por el abaratamiento que supone. Esto implica una combinación de *hardware* y *software* que permita la conexión de un cable estándar *Ethernet* y el procesamiento de la información que transmita.
    - En las aplicaciones de los relés, especialmente en los sistemas de prueba de aviónica, se suelen conectar unidades muy valiosas que operan con una gran precisión, en consecuencia, hay que evitar que señales indeseadas alcancen los dispositivos. Durante la conmutación de los relés, concretando el caso de configuraciones de relés con múltiples posiciones (*SPQT*), es imprescindible asegurar que no aparece señal alguna en posiciones indebidas, tanto por ruido proporcionando un buen aislamiento, como por cortocircuitos incorrectos garantizando la integridad funcional.
    - El sistema está proyectado para la ejecución en *VXI*; sin embargo, la versatilidad en cuanto a la operación en un *mainframe* independiente, dedicado únicamente al gobierno de los módulos de relés, supone un gran valor añadido porque facilita la sustitución de un *mainframe VXI*, económicamente costoso, por uno de diseño propio, más asequible y con la misma funcionalidad.

1.4 Metodología

El conjunto de procedimientos llevados a cabo para alcanzar el objetivo de este proyecto técnico gozan de gran importancia, ya que marcarán el rumbo del diseño y determinarán las prestaciones y el comportamiento del sistema de *switching* modular.

La metodología de diseño elegida es comúnmente denominada *Top-Down*. La estrategia de procesamiento de información *Top-Down* consiste en examinar detenidamente el proyecto (con un alto nivel de abstracción), desarrollar la idea y aumentar el nivel de detalle hasta precisar los componentes primarios del diseño. De forma que el diseño inicial es ramificado en planteamientos más reducidos que a su vez se pueden continuar dividiendo en elementos más pequeños y más asequibles. Este proceso resulta en la segmentación de la idea clave en módulos hasta llegar a elementos concretos, lo que facilita la verificación del sistema, se posibilita su simulación y deriva en una considerable reducción de la posibilidad de inclusión de errores en el proyecto técnico.

El diseño *Top-Down* se ajusta aún más a este proyecto debido a que se trata de un sistema de instrumentación modular, y por tanto se organiza de manera fraccionada. Los módulos en los que estará subdividido el sistema serán funcionalmente independientes.

Esta metodología es aplicable independientemente a los aspectos *hardware* y *software* del proyecto técnico; sin embargo, es necesario distribuir el proyecto en varias porciones claramente diferenciables que requieren de procedimientos específicos, tratadas en sus secciones correspondientes.

Si bien el desarrollo del sistema puede realizarse diseñando múltiples facetas al mismo tiempo, es favorable asentar primero una base con la implementación del *hardware*, de manera que el diseño del *software* se adapte en gran medida al diseño del *hardware*. Una vez que el hardware se encuentre completamente elaborado, cualquier modificación posterior supondría un alto coste económico y una prolongación del tiempo de desarrollo.

Capítulo 2. Sistemas *VXI* actuales

Una de las misiones principales de este proyecto consiste en sustituir y/o compatibilizar el sistema *VXI* por una tecnología más económica que ofrezca los mismos servicios.

A día de hoy, el *bus VXI* se ha convertido en un sistema de arquitectura abierta muy exitoso y se realizan multitud de pruebas sobre equipos para demostrar su fiabilidad y validarlos. Estos ensayos y mediciones cobran mayor importancia en sistemas y elementos precisos, de alto rendimiento y de valor económico elevado (como en aviónica, transportes masivos terrestres y marítimos, etc.). Dadas las características de los sistemas *VXI*, gozan de especial interés en las verificaciones de equipos electrónicos de aeronaves.
Generalmente la aviónica militar se somete a pruebas y medidas a tres niveles de montaje:

- Prueba a nivel de módulo (o tarjeta de circuito impreso de montaje).
- Prueba a nivel de unidad (o caja).
- Prueba a nivel de sistema.

Los equipos especiales de pruebas son usados para suministrar unos *tests* meticulosos necesarios a estos tres niveles. Diseñar un sistema capaz de validar aviónica avanzada militar es extremadamente costoso y además, implementar un diseño con las restricciones programadas por programas militares es una tarea ardua.

Muchas empresas dedicadas al ámbito militar se han aprovechado de los atributos del *bus VXI* para resolver con éxito las aplicaciones de prueba más exigentes:

- Volver a hospedar un sistema heredado de pruebas militares. [7]
  Hay multitud de sistemas militares anticuados que necesitan modernizarse. Los diseñadores de sistema evolucionaron desde racks autónomos a instrumentación modular *VXI* como parte de una solución, generalmente combinados con instrumentos no modulares. Actualmente las necesidades militares de verificación de unidades se fundamentan en un sistema de demostración de soporte de combate global, ágil y rápido (*ARGCS*), proyectado para ejecutar un conjunto de programas de pruebas (*TPS*).
  Los instrumentos *VXI* fueron escogidos por numerosas razones, entre las que destacan:
    - Mantenimiento, el tiempo medio utilizado para la reparación de sistemas *VXI* es relativamente reducido. La mayoría de los instrumentos de *VXI* se conectan al chasis de manera que son fácilmente accesibles por la parte frontal para la retirada del cableado y su extracción por módulos, y los que no muestran estas características, pueden ser fácilmente desenchufados y repuestos sin esfuerzo. Esto lo convierte en una plataforma idónea para realizar testeos y mediciones de elementos militares críticos para una misión.
    - Velocidad de transferencia de datos, la mayoría de aplicaciones no están limitadas por la rapidez de la computadora para procesar bloques de datos, sino por el tiempo necesario para trasladar y manejar la información, o la velocidad a la que los instrumentos pueden generar o aceptar dichos datos. Por ello, una alta velocidad de transferencia de datos (como la de los sistemas

VXI) supone una propiedad fundamental para servicios de valoración de *hardware*.

- Diversidad de los instrumentos, hay una gran variedad de instrumentos (tanto analógicos como digitales) y una vasta pluralidad de fabricantes que pueden suministrarlos.
- *Hardware* de medición de alto rendimiento, *VXI* cuenta con cuantiosos dispositivos que conceden una gran precisión en la evaluación de señales en un sinfín de modalidades.
- Buena especificación del *hardware* y *software* de integración, las características de alimentación y del sistema de refrigeración se encuentran correctamente determinadas, lo que unido a herramientas de *software* (como *IVI, VISA* y *VXI Plug&Play*) permiten una integración predecible y directa.
- Flexibilidad y reconfiguración, la flexibilidad del *VXI* permite la estandarización de componentes para un gran abanico de canales encargados de multitud de pruebas de alto rendimiento. En caso de que un alto rendimiento no sea estrictamente necesario, permite hacer uso del equipo en pequeños sistemas cuando no es requerido en *tests* complejos. Por ejemplo, un sistema de 1024 canales puede ser configurado en 8 pequeños sistemas y aprovechado indistintamente cuando no es utilizado en un test de dimensiones considerables.
- Compactibilidad del rack, *VXI* permite la inserción de hasta 12 instrumentos en el chasis, que se podrían convertir en aún más si tenemos en cuenta que estos dispositivos pueden incorporar módulos *mezzanine* en sus *carriers VXI*.
  Además estos chasis incorporan componentes en el que pueden ser eliminados de la estructura de los instrumentos al ser redundantes (como piezas de alimentación, conectores, sistemas de refrigeración, etc.).
- Integración, *VXI* provee una solución altamente integrada entre el acondicionamiento y la medición.

- Construir un sistema de test con instrumentos sintéticos (*SI*). [7]
  Los sistemas sintéticos consisten en una serie de dispositivos de medición que incluyen:
  - *RF Down-converters* con oscilador local, que convierte una señal digital real centrada en una frecuencia intermedia a una señal compleja centrada a frecuencia nula.
  - *RF Power Meters*, que se encargan de medir la potencia de una señal de RF.
  - *LF Signal Conditioner*, que manipula una señal analógica de baja frecuencia de tal manera que cumpla unos requisitos prefijados.
  - *Analog to Digital Converters* (de alta velocidad y de banda estrecha), que convierten señales analógicas a señales digitales.
  - *RF Synthesizer*, que genera cualquier rango de frecuencias.
  - *Arbitrary Waveform Generator*, que permite generar señales eléctricas.
- Un repuesto sintético para un instrumento obsoleto. [7]
  Es evidente que un instrumento *VXI* puede coexistir y operar sin problema con otros instrumentos *VXI* distintos independientemente de su fabricante; más allá, otro tipo

de dispositivos pueden trabajar en un chasis *VXI* con instrumentos *VXI* (como módulos *PXI*, tarjetas *mezzanine*, módulos *VME*, etc.).

La aplicación de la metodología del *VXI* en la detección de fallos en aviónica ha resultado en numerosos beneficios. La disponibilidad de numerosos dispositivos estandarizados ha permitido una acentuación de la eficacia y la eficiencia en la verificación y mantenimiento de equipos electrónicos de aeronaves militares. [30]

Aunque los sistemas *VXI* suponen una alta inversión, cómodamente amortizable en aplicaciones militares, también tienen cabida en usos comerciales por su arquitectura modular abierta que respalda todo tipo de aplicaciones de medida y ensayo. Los usuarios tienden a estandarizar un ordenador común, entornos de desarrollo de *software* y consolidar *hardware* de prueba para explotar su adquisición (en términos económicos y de progreso).

Los fundadores del consorcio de *VXI* aplicaron sus conocimientos combinados como industria y determinaron el futuro de la arquitectura de acuerdo a las demandas de aplicaciones de test de los 20 años posteriores, cuando son claves el tiempo de reparación, la capacidad de configuración, la expansibilidad, la densidad, la fiabilidad y el mantenimiento.

Los instrumentos *VXI* continúan experimentando perfeccionamientos incrementales, con contribuciones significantes en el área mecánico-estructural y de acondicionamiento de señal, que suponen importantes contribuciones a la integración y al rendimiento del sistema.

VXIbus se encuentra a mitad de su vida previsible. Todo este tiempo ha seguido mayormente un camino dedicado a su construcción y desarrollo, y ha logrado encontrar los componentes adecuados para proporcionar una gran calidad a los usuarios. Los siguientes pasos en la evolución de estos sistemas son etapas de refinamiento, con una implementación consolidada, reducciones en el coste y mejoras graduales. Las tecnologías de *Software* (como *VXI Plug&Play*, *ActiveX*, etc.) facilitan una correcta integración de estos avances en la implementación de aplicaciones de prueba.

La tecnología *VXI* acompaña a las tendencias de incorporación e integración de tecnología, permitiendo su consolidación, reconfiguración del *hardware* y reutilización de los componentes de *hardware*. Aunque la prosperidad del *VXI* se manifieste en implementaciones en el sistema y el estándar *VXI* mantenga un estado de constante evolución, conserva la compatibilidad de todas sus facetas.

2.1 Fundamentos del *bus VXI*

VXI está especialmente diseñado para aplicaciones de medidas y *tests* informatizadas. Como indica su nombre está basado en el *VMEbus* (consultar anexo A.1.1), un *bus* de ordenador basado en mensajes muy popular en aplicaciones industrializadas por ordenador. Los diseñadores se basaron en este *bus* y añadieron señales para aumentar la precisión en el tiempo, sincronizaciones entre módulos en el *bus* y recursos para su manejo y aumentaron el tamaño de las tarjetas para proporcionar una capacidad de medición mayor.

El *VXIbus* no fue diseñado para remplazar ningún estándar, sino como una herramienta adicional para solucionar problemas de test o adquisición de datos. Para ello se definieron varios métodos de comunicación, lo que compatibiliza a este sistema con el *VMEbus*, *GPIB*, soluciones portátiles independientes, etc.

Los sistemas *VXI* pueden ser configurados de infinitas formas. Los módulos de instrumentación del *VXI* residen en un chasis o *mainframe VXI* (en cualquier posición excepto la ranura correspondiente a la tarjeta controladora), que puede contener hasta 12 módulos. Además, el mainframe se ocupa proporcionar la potencia, la refrigeración y la comunicación apropiadas para que los instrumentos puedan desempeñar sus funciones. [22]

En todo sistema *VXI* coexisten dos ocupaciones de control, en ocasiones coincidentes en el mismo elemento, que posibilitan el correcto funcionamiento [54]:

- El *slot 0* tiene el compromiso de controlar el *backplane*, es decir, la placa base que actúa como red de distribución de energía y señales a todos los módulos, y arbitrar el movimiento de datos a través de este e incluir señales externas (como una señal de reloj). Además de realizar estas acciones también debe ocuparse de sus funciones.
- El administrador de recursos, un programa de ordenador, configura los módulos para posibilitar su adecuada operación y una vez que esta comienza, el administrador de recursos ya no se involucra.

En caso de que el sistema necesite más *slots* de los proporcionados por un chasis, es posible conectar un segundo chasis. El ordenador que controla el sistema y permite la comunicación con un operador, bien debe radicar en el chasis o en su lugar tiene que estar conectado al chasis mediante una interfaz *IEEE 488* o una interfaz de mayor velocidad.

Una característica poderosa del sistema consiste en que los instrumentos pueden comunicarse entre ellos rápidamente. La especificación de *VXIbus* determina dispositivos basados en registros (como análogos a los instrumentos *VME*, su programación requiere de la escritura y lectura de los registros del instrumento) y dispositivos basados en mensajes (capaces de conseguir un nivel de comunicación más elevado).

Para conseguir diversas ventajas en el ámbito de la instrumentación modular, la arquitectura de los sistemas *VXI* presenta varias capas autónomas [32]:

- La última capa es la capa de *hardware*, constituida por instrumentos de medida.
- Por encima de esta capa se encuentra el *bus* de control, que se encarga de conectar los múltiples dispositivos a un dispositivo de gobierno.
- Subiendo una capa, el instrumento de medición es dominado a través de un controlador o *driver* del dispositivo y/o una interfaz de programación de aplicaciones (*API*). Estos elementos interactúan con las herramientas de configuración y depuración, de manera que ayuda al usuario en el desarrollo del control de la aplicación.
- Finalmente, estas capas se combinan en un ordenador, en el programa de usuario de la aplicación de test, compuesta por simplemente una aplicación o un complejo sistema de manejo.

2.2 Especificaciones *VXI*

El sistema *VXI* está basado en una industria abierta, es decir, es posible afrontar las elecciones de los componentes del sistema teniendo en cuenta únicamente las necesidades, pese a que los elementos finalmente seleccionados pertenezcan a distintos vendedores. Los estándares abiertos también aseguran que una vez que el sistema es construido, la inversión continuará generando beneficios en el futuro.

El estándar *VXI* determina las características eléctricas del *bus*, las especificaciones mecánicas, de potencia, de refrigeración y de interferencias electromagnéticas de los módulos *VXI*. Estas distinciones aseguran que los módulos de diferentes fabricantes puedan funcionar juntos cuando trabajan en el mismo sistema.

La fundación de instrumentos virtuales intercambiables (*IVI*) se basa en estas especificaciones para abordar el sistema *VXI* como un todo, con la meta de que el usuario final pueda obtener un producto capaz de ponerse en funcionamiento en poco tiempo, introduciendo paneles frontales sencillos, *driver*s de instrumentos y rutinas de instalación que siguen las directrices del fabricante, para extraer el máximo provecho de las capacidades de la instrumentación y hacer de la programación una tarea lo más sencilla posible. (Ver anexo A.1.2). [22]

2.3 Infraestructura integrada de *software*

El *software* es uno de los elementos más importantes a considerar en un sistema *VXI* y actualmente, los desarrolladores de sistemas se enfrentan a la necesidad de un marco de *software* integrado. Esta infraestructura debe simplificar las complejidades que entraña la integración múltiple de instrumentos de medida y ensayo, en un único sistema que suministre interfaces para todos los dispositivos de entradas y salidas. Igualmente debe proporcionar herramientas de desarrollo para velozmente configurar, construir, implementar, mantener y aumentar el rendimiento, con soluciones de control y medición de bajo coste. Este marco de *software* integrado está obligado a conceder una conectividad sin fisuras a la constante evolución del sistema y gracias a ello los distribuidores de productos entregan estos rápidamente, con mayor calidad y menor coste de desarrollo y producción. Asimismo, mantiene una estrategia en la que los componentes de la infraestructura son independientes, a la vez que estrechamente ligados, lo que acelera la implementación de sistemas de test y facilita la adaptación a los requisitos.

La fundación *Interchangeable Virtual Instruments* (*IVI*) ofrece una integración del sistema asequible y determina varias infraestructuras de *software* interoperables a elegir según los requisitos, de manera que proporciona todos los mecanismos necesarios para construir sistemas que satisfagan las necesidades. Aunque parece una meta ardua de conseguir, la fundación *IVI* lo ha logrado definiendo e implementando condiciones para todo el sistema de pruebas, introduciendo *software*, comunicación de entradas y salidas, *driver*s, paquetes de instalación, ordenadores e interfaces. El *software* de *VXI* usa la última tecnología que ha

evolucionado para la programación de instrumentación y simplifica la planificación de tareas, lo que convierte a *VXI* en un sistema fácil de usar. [51]

Las decisiones *software* no sólo afectan de manera global al rendimiento y la capacidad del sistema, también influyen en el tiempo de desarrollo, mantenimiento, productividad y reutilización del *software* para futuros proyectos. Para empezar, hay que determinar el marco o plataforma de trabajo.

Una vez elegido el marco en el que se desarrollara el *software* (teniendo en cuenta las exigencias de la aplicación de testeo), entonces la denominación de este marco otorga la información pertinente para escoger el resto de componentes *software* mediante una especificación definida. Es decir, dada una plataforma de trabajo, es necesario elegir un *PC* y sus componentes necesarios para el control del *hardware*, un sistema operativo, una interfaz de comunicación, entornos de desarrollo compatibles, documentación necesaria y soporte para la instalación, un panel frontal, un mainframe *VXI*, un controlador del sistema para el *slot 0* e instrumentos *VXI* compatibles con esta plataforma.

Los instrumentos *VXI* son controlados por una serie de rutinas de control dentro de la plataforma de trabajo. Cada una de estas rutinas corresponde con una operación programada en un protocolo determinado. Mediante un controlador o *driver* se elimina la necesidad de aprender este protocolo, de manera que el control del instrumento sea una tarea sencilla.

Los *driver*s son construidos en entornos de desarrollo de aplicaciones de acuerdo a unas arquitecturas, en concreto para los instrumentos *VXI*, *Plug & Play* o *Interchangeable Virtual Instrument* (*IVI*) [51] [11]. Ambas arquitecturas usan *Virtual Instrumentation Software Architecture* (*VISA*) para proporcionar una comunicación del instrumento independiente del *bus* y de la plataforma.

2.3.1 Arquitectura de software de instrumentos virtuales (*VISA*)

Las especificaciones de *VISA* (*Virtual Instrument Software Architecture*), bajo la supervisión de la fundación *IVI*, ofrecen un estándar común para el desarrollo de programas de *software* de diversos fabricantes, incluyendo controladores de instrumentos. Incorpora una interfaz, *VISA*, de programación y comunicación entre el *hardware* y los entornos de desarrollo, proporcionada por una capa aislada de *software*. Participa en las tareas de configuración, programación y resolución de problemas de sistemas de instrumentación, constituidos por interfaces *MXI, GPIB, VXI, PXI, Serie, Ethernet* y/o *USB*.

VISA determina una arquitectura repleta de recursos (objetos que hacen referencia a instrumentos) que sintetizan la funcionalidad del dispositivo. Cada recurso puede proveer servicios especializados a aplicaciones u otros recursos, lo que demuestra la existencia de una gran consistencia en la operación de recursos *VISA* conseguida mediante una interfaz extensible y definida con precisión. [49] [50]

Un recurso *VISA* deriva su interfaz a partir de una plantilla que proporciona servicios estándar para el recurso, lo que incrementa la capacidad de reutilización, test y mantenimiento del recurso. Los servicios fundamentales son los siguientes:

- Crear y eliminar sesiones (canales de comunicación), controla su ciclo de vida.
- Modificar y recuperar características de un recurso, también llamados atributos.
- Concluir operaciones en cola.
- Restringir el acceso al recurso.
- Realizar servicios básicos de comunicación.

VISA también cuenta con un administrador de recursos (*VISA Resource Management*), que es en sí mismo un recurso, capaz de otorgar servicios (acceso y búsqueda de recursos), conectividad a todos los recursos *VISA* y control y acceso de aplicaciones a los recursos.

2.3.2 Controladores de instrumentos (Drivers)

Los controladores de instrumentos deben estar estandarizados de tal manera que los *driver*s suministrados por los distintos fabricantes puedan ser usados en el mismo sistema. Además, deben mostrar consistencia en su arquitectura, diseño y uso, así como presentar toda la documentación necesaria.

Los *driver*s se ocupan de la comunicación de entrada y salida a bajo nivel, por lo que al desarrollar un sistema no es necesario enfrentarse a los protocolos de entrada y salida, ya que los *driver*s incluyen:

- Librerías de enlace dinámicas (.dll o .sl), el código fuente *ANSI* y un fichero de panel de funciones (.fp). Estas herramientas permiten utilizar un lenguaje de alto nivel (como funciones C) en la aplicación.
- Programas ejecutables con paneles frontales interactivos, es decir, una interfaz gráfica para controlar un instrumento ya integrado en el sistema.
- Un archivo de base de conocimientos *ASCII* que describe todas las especificaciones del dispositivo (mecánicas eléctricas e información del entorno).
- Un fichero de ayuda suministra información sobre las librerías de funciones C, el panel frontal, ejemplos de programación y una visión de conjunto del instrumento.

El control de instrumentos *VXI* se enfrenta a dos paradigmas, dependientes del tipo de instrumento [21]:

- El control basado en mensajes, en el que el dispositivo es gobernado mediante comandos de alto nivel *ASCII* y cadenas de datos que son comunicadas siguiendo un estándar e interpretadas por cada instrumento de una manera determinada.
- El control basado en registros, en el que el instrumento es manejado mediante lecturas y escrituras de los registros binarios individuales.

Un buen *driver* debería implementar las funciones básicas *Initialize, Close, Reset, Self-Test, Error Query y Revision Query*, y una función de mensajes de error (*Error Message*) en caso de que se desarrolle en lenguaje C.

Existen dos caminos hacia una correcta interacción con el dispositivo *VXI*, a través de los controladores *VXIplug&play* y mediante los *driver*s de instrumentos *IVI*. Según las necesidades es conveniente el uso de uno u otro:

- El *driver VXI Plug&Play* es un controlador de instrumentos clásico y con capacidades limitados, pero relativamente simple de implementar.
- El d*river IVI* es un controlador de instrumentos más complejo y con competencias más notables.

2.3.2.1 La Fundación de Instrumentos virtuales intercambiables (IVI)

2.3.2.1.1 Metas de la Fundación *IVI*

La Fundación *IVI* es un grupo de compañías, integradores de sistemas, y fabricantes de instrumentos que colaboran para elaborar y promover las especificaciones para la programación de dispositivos. Comparten el compromiso de garantizar el éxito del desarrollo de sistemas de prueba mediante una tecnología de control de instrumentos abierta y potente. Fomenta el uso de interfaces de programación de aplicaciones estándares, la intercambiabilidad, el rendimiento en la ejecución y la simulación, para afrontar las dificultades de las tareas de programación. [12] [51]

El consorcio fue fundado principalmente para promover las especificaciones de programación de instrumentos de test de acuerdo a las siguientes metas:

- Intercambiabilidad del *hardware*:
  - Para facilitar la tarea de sustitución de instrumentos en un sistema por otros instrumentos similares.
  - Preservar el *software* de medida de cara a la obsolescencia de los dispositivos.
  - Simplificar la reutilización del código de test desde la validación del diseño hasta la producción del *test*.
- Calidad:
  - Aumentar la calidad de los *driver*s.
  - Establecer patrones de prueba y verificación de controladores.
- Interoperabilidad del *software*:
  - Proporcionar un acceso estándar a las capacidades del *driver* como el almacenamiento en caché del estado o la comprobación de rango
  - Suministrar un entorno de trabajo que permita al usuario la integración sencilla independientemente del fabricante de instrumentos.
  - Simular dispositivos y desarrollar *software* cuando estos no están disponibles físicamente.

- Otorgar consistencia al control del instrumento en entornos de programación populares.

2.3.2.1.2 Arquitectura de los controladores *IVI*

La eficacia de la instalación y soporte de los sistemas de *test* es entorpecida por el alto coste del desarrollo y mantenimiento del *software*, y por la veloz evolución de la tecnología. La fundación *IVI* aborda estos problemas mediante una nueva tecnología de *driver*s.

La arquitectura *IVI* de los controladores de instrumentos se basa en el lenguaje C y hace uso de los estándares definidos por *VXIplug&play Systems Alliance*, conservando requisitos como el formato del panel de funciones y de los *subarchivos*, y numerosas funciones (*Initialize, Reset, Self-Test* y *Close*). Otras funciones y especificaciones, no obstante, son redefinidas en las especificaciones *IVI* (como el *Error Handling*, el formato de los nombres, etc.). [11]

Los controladores *IVI* se comunican directamente con el dispositivo *hardware* o actúan como una capa intermedia a otro controlador *IVI*. Se dividen en:

- Controlador específico *IVI*. Es un *driver* que contiene la información necesaria para controlar un instrumento en concreto o una familia de instrumentos, mediante una comunicación directa con el dispositivo. A su vez , están constituidos por:
  - *Drivers* específicos de clase compatible, que cumplen con las especificaciones de clase *IVI*. Exportan *APIs* ya definidas e incorporan características que posibilitan el intercambio de instrumentos, por lo proporcionan una solución al deseo de la independencia del *hardware*. Dependiendo de la clase de *API* se categorizan en:
    - Controlador específico de clase combatible con *IVI-COM*.
    - Controlador específico de clase combatible con *IVI-C*.
    - Controlador específico de clase combatible con *IVI.NET*.
  - *Drivers* específicos personalizados, que no pueden ser utilizados para intercambiar instrumentos porque exportan una *API* individualizada. Se usan en dispositivos especializados.
- Controlador de clase. Es un *driver* que permite a los usuarios intercambiar instrumentos. Exportan un *API* que se ajusta a las especificaciones y se comunican con los dispositivos mediante una llamada a un controlador específico de clase compatible.

Con el fin de lograr la intercambiabilidad la Fundación creó especificaciones *IVI* de clase, que determina las capacidades básicas y extendidas de la clase para los tipos de instrumentos más populares (multímetros digitales, osciloscopios, generadores de funciones, fuentes de alimentación, tarjetas de relés, medidores de potencia, analizadores de espectros, etc.). Los miembros de la Fundación compartieron componentes de *software* que ofrecen servicios a los *driver*s y sus usuarios, de manera que mantienen características comunes. [11]

Todos los controladores *IVI* se comunican con el *hardware* mediante una librería *VISA I/O* (de entradas y salidas), utilizable en *bus*es *GPIB* y *VXI*.

2.3.2.2 Controladores de instrumentos VXI Plug&Play

VXIplug&play Systems Alliance fue fundada por miembros que compartían una voluntad común, la consecución por parte de los usuarios de la implementación en sistemas *VXI* de diversos fabricantes. Más allá del alcance de las especificaciones del *bus VXI*, la alianza logró acrecentar la facilidad de uso y aplicar normas y prácticas, en el *hardware* y *software*, que intensificaron la interoperabilidad. [50]

En 2002, *VXIplug&play Systems Alliance* tomó la decisión de anexionarse a la Fundación *IVI*, y un año más tarde se fusionó otorgándole el poder a la Fundación, que actualmente vela por las especificaciones de *VXIplug&play*.

VXI Plug&Play es un estándar, creado por *VXIplug&play Systems Alliance*, especialmente definido para el desarrollo de controladores de instrumentos *VXI*, con la finalidad de entregar al usuario un *driver* familiar y sencillo de usar con cualquier equipo de *VXI*, indistintamente del fabricante. Se asegura de que *VXI* permanezca siendo una tecnología abierta y producida por numerosos fabricantes a nivel de sistema y módulo.

Los controladores de *VXIplug&play* se basan en *VISA* y ofrecen librerías para el control programado, usado en una gran variedad de entornos de desarrollo de *software* de alto nivel. La especificación de la arquitectura del controlador usa dos modelos [52] [53] [54]:

- Modelo de interfaz externa del controlador, que muestra cómo interactúa el controlador con el resto de *software* del sistema. Está formado por:
    - Cuerpo funcional, el núcleo del controlador del dispositivo.
    - Interfaz programática del desarrollador, el mecanismo de llamada al *driver* desde un programa de *software* de mayor nivel.
    - Interfaz interactiva del desarrollador, normalmente gráfica, que ayuda a los desarrolladores de *software* a entender que desempeña cada función del controlador y cómo usar la interfaz programática del desarrollador para llamar a cada función.
    - Interfaz *VISA I/O*, el medio a través del cual el *driver* se comunica con el *hardware*.
    - Interfaz de subrutina, mediante la que el controlador puede llamar a otros módulos de *software*.
- Modelo de diseño interno del controlador, que determina como es organizado internamente el *software* de un instrumento. Se estructura por niveles:
    - En el nivel más alto se encuentran las funciones de la aplicación, un conjunto de funciones de alto nivel que realizan las operaciones de medida y prueba. Representan las capacidades del instrumento.
    - A un nivel más bajo se sitúan las funciones de componentes, que controlan un área determinada del instrumento y se suelen descomponer en funciones más simples. Se dividen en cuatro categorías:
        - Función de Inicializar (*Initialize*), que establece la conexión con el instrumento y crea una sesión con el instrumento.

- Funciones de Clases de capacidad (*Capabilty classes*), que clasifican las funciones de acuerdo a las capacidades y tras su ejecución se devuelve el valor de finalización y el valor de la acción solicitada. En el nivel más alto de estas funciones se ejecuta una acción completa.
- Funciones de utilidad (*Utility*), que realizan una gran variedad de operaciones. Como por ejemplo*: Reset, Selft-Test, Error Query, Error Message, Revision Query,* etc.
- Función de Cerrar (*Close*), que termina la sesión inicializada con el instrumentos y desasigna los recursos del sistema asociados a dicha sesión.

El sistema requiere de una funcionalidad por parte del *driver* muy simple, por lo que no es necesario complicar el desarrollo de un controlador para conseguir un buen funcionamiento. *VXI Plug&Play* representa una alternativa más adecuada para este proyecto.

2.4 Sistemas de *switching VXI*

Los componentes básicos e imprescindibles para desarrollar sistemas de *switching* son relés. El tipo de relé más popular es el electromecánico, aunque existen más clases de relés (relés de estado sólido, relés de corriente alterna y relés de láminas).

En el caso de aplicación a sistemas de prueba y medida, el relé más conveniente es el relé electromecánico. Un relé electromecánico funciona como un interruptor gobernado por un circuito eléctrico en el que, al pasar corriente eléctrica por una bobina, un núcleo de hierro se magnetiza convirtiéndose en un imán, que a su vez atrae a un inducido (normalmente una varilla de metal) hasta hacer contacto con otra parte del relé y crear un camino alternativo. Hay varios tipos de relés electromecánicos, pero dada la aplicación del sistema marcada por el ámbito de trabajo de cada módulo de relés, se emplean los relés de armadura.

Los relés de tipo armadura son los más antiguos y siguen siendo los más utilizados. Un electroimán es excitado y ocasiona la basculación de una armadura, provocando un cortocircuito entre la posición común (*COM*) y el contacto normalmente abierto (*NO*), y dejando un abierto entre el común y el contacto normalmente cerrado (*NC*).

A menudo, el objetivo del diseño de instrumentos de *switching VXI* consiste en satisfacer las especificaciones utilizando el menor número de relés posibles; sin embargo, en el ámbito de sistemas de prueba y medida prima la consecución del diseño de unos dispositivos veloces y de alto rendimiento.

La clave para lograr instrumentos rápidos y ágiles en el soporte de aviónica de combate se encuentra en la flexibilidad. En cuestión de dispositivos de conmutación, la esencia de esta flexibilidad radica en la implementación de un conjunto de relés que puedan ser reconfigurados en otras distribuciones. El desarrollo de sistemas de conmutación o *switching* necesita relés y pistas adicionales en los módulos, por lo que resulta imprescindible más espacio, de ahí que muchos dispositivos de *switching VXI* ocupen dos *slots*.

Capítulo 3. Diseño del *hardware*

3.1 Requisitos *hardware*

Los objetivos del presente proyecto describen de manera muy general las metas que debe alcanzar el sistema, pero existen diversos requisitos específicos puntuales del *hardware* que determinan el camino por el que se ha de implementar el sistema.

Las especifcaciones mecánicas de cada uno de los módulos de relés se encuentran influenciadas por el tamaño C (ver anexo A.1.2.1) de *VXI*. Puesto que la altura es relativamente escasa, se pretende elaborar un diseño que permite ocupar cada ranura con al menos dos módulos de relés. Si tenemos en cuenta numerosas soluciones de instrumentos *VXI*, distribuyen la funcionalidad en pequeños módulos de unos 11 cm de alto y diversas profundidades de hasta 34 cm, con un ancho de unos 2 cm, de manera que un instrumento con 6 módulos ocupa dos ranuras de *VXI*. Estos productos suponen un buen recurso de inspiración mecánica, por lo que ajustarse a estas medidas simplifica de forma idónea las características mecánicas. La figura 1 ilustra la disposición de 6 módulos en dos *slots VXI*.

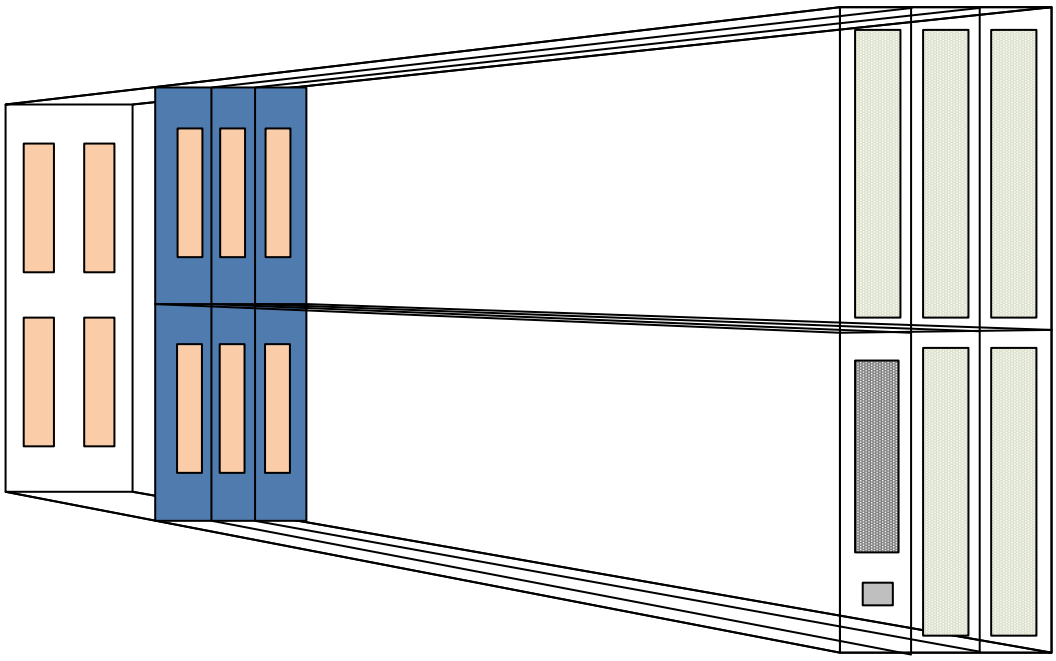


Figura 1. Instrumento de 6 módulos establecido en dos ranuras *VXI*

Las tarjetas *VXI* son suministradas con diversas alimentaciones, pero solo es realmente concerniente la alimentación de 5V que emplearán todos los módulos implementados, por las características de los relés. Aunque para una compatibilidad completa con *VXI* sería necesario el desarrollo de un chasis o cableado, que conectase los módulos de relés a los conectores del *backplane* del *VXI*, en este proyecto no se profundiza en la solución mecánica de este problema.

Estas especificaciones atañen diversos elementos:

- El módulo de relés *IND2002A* muestra las siguientes singularidades:
  - 12 relés (CH) *AZ944-1C-5DE* o con características eléctricas similares o superiores, en una configuración *SPDT* (*Simple Pole Double Through*), es decir, un único camino común (*COM*) y dos posiciones conmutables, denominadas *NO* (normalmente abierta) y *NC* (normalmente cerrada).
  - Conector comercial *GMCT41M* con un conexionado hacia los relés establecido por el *wirelist* apropiado (anexo A.5.1.1).
- El módulo de relés *IND5002* presenta estas distinciones:
  - 50 relés *EC2-5NJ* o con especificaciones eléctricas semejantes o mejores, en una configuración *SPDT*.
  - Conector comercial *DIN 41612 004778* conectado de manera que asigne las señales de los relés a los pines siguiendo el *wirelist* conformado (anexo A.5.1.2).
- El módulo de relés *IND5003* presenta estas distinciones:
  - 52 relés *EC2-5NJ* o con especificaciones eléctricas parecidas o superiores, en una configuración *SPQT*, o sea, un terminal común y cuatro posiciones conmutables (1, 2, 3 y 4).
  - Conector comercial *DIN 41612 004778* acoplado de modo que relacione las señales de los relés a los pines acogiéndose a su correspondiente *wirelist* (anexo A.5.1.3).

3.2 Procedimientos específicos del diseño *hardware*

El diseño del *hardware* asistido por ordenador o *CAD* (*Computer Aided Desing*) utiliza tecnología gráfica para facilitar el diseño y su comprensión. A día de hoy, esta técnica de diseño es indispensable para proyectar e implementar circuitos electrónicos. [33]

En el marco de creación del *hardware* existe un amplio rango de herramientas *CAD* que se encuentran a lo largo de todo el proceso de diseño:

1. En la etapa de especificación de la idea, mediante un sistema de bloques y esquemáticos, que consiste en el uso de esquemas que representan la disposición del sistema de manera sintetizada, además de incorporar información sobre líneas de datos, componentes, conectores…
2. A continuación en la simulación, ya sea funcional, digital, eléctrica o por eventos. Aunque las herramientas *CAD* proveen de una interfaz gráfica muy poderosa que asiste en el proceso de creación, su verdadero potencial se encuentra en la posibilidad de simular el circuito. De este modo, es posible verificar el funcionamiento del sistema antes de su fabricación, evitando así los costes derivados de un diseño deficiente y disminuyendo la demora en el diseño del sistema.
   En este caso no se ha procedido a la simulación del *hardware* debido a la relativa sencillez del mismo y a la dificultad para enlazar componentes tecnológicamente

recientes, autónomos y muy específicos en los *CAD* (véase elemento *XPort* en el Anexo A.2.3.1). Por lo que es más apropiado reproducir la respuesta de los elementos determinantes en otra fase de desarrollo.

3. Por último, en el diseño de circuitos impresos, para el modelado geométrico (tanto de la placa como de los componentes y sus huellas) y el trazado de pistas que enlazan unos componentes con otros. Una vez concluido el desarrollo del circuito impreso las herramientas *CAD* permiten la generación de los documentos necesarios para la fabricación del prototipo.

Existen multitud de herramientas *CAD* para llevar a cabo estos propósitos; sin embargo, llevando a cabo una buena comparativa de diversas plataformas de desarrollo y teniendo en cuenta la finalidad de la herramienta, se alcanza una conclusión, *Altium Designer*, ya que es un *software* especialmente orientado al diseño de placas de circuito impreso. Otras herramientas *CAD* (como *Proteus* o *MultiSim*) se centran más en la simulación (aspecto que no es necesario en este proyecto) o han quedado un poco obsoletas (como *OrCAD* por su herramienta de rutado).

3.3 *Altium Designer*

Altium Designer es una herramienta automatizada de desarrollo electrónico muy potente que ofrece soluciones de diseño completas desde una misma plataforma. Además de incorporar diseño de *PCB (Printed Circut Board)* y esquemáticos, incluye simulación de circuitos, análisis de integridad de señal, integración de *FPGA* y *HDL*, herramientas de *software* y compiladores, listas de materiales, etc.

En la implementación de un proyecto desde cero, como es el caso, es necesario completar unos esquemáticos de manera precisa y detallada, ya que el diseño de una *PCB* constituye la versión de fabricación de estos diagramas. El proceso de construcción de esquemáticos es realizado en un editor que otorga multitud de capacidades de interacción. Usa dos tipos de objetos, gráficos (líneas, curvas, cuadros de texto, imágenes, etc.) y eléctricos (prediseñados para crear y representar el circuito) para facilitar la comprensión del circuito. *Altium Designer* integra numerosas librerías de objetos eléctricos que suponen un ahorro significativo del tiempo de desarrollo de los componentes del circuito; sin embargo, también es posible elaborar los elementos del circuito mediante herramientas incluidas en la librería de esquemáticos.

El editor de esquemáticos presenta varias formas para conectar objetos eléctricos entre ellos:

- Cables, es la conexión básica y realiza una conexión eléctrica entre dos puntos.
- Redes, es una conexión eléctrica bien representada por un identificador o un nombre único o en su lugar por una etiqueta de red en un cable.
- Puertos de alimentación, que proporcionan un modo fácil de conectar componentes a terminales de alimentación
- *Bus*es, que consisten en agrupaciones de cables o señales.

- Conjuntos, que asocian cables, *bus*es y otras señales.

A menudo, la implementación del esquema del circuito en un solo diagrama resulta una tarea impracticable o difícil de presentar, por lo que mediante el uso de los elementos de conexionado y la capacidad de *Altium* de crear proyectos *multi-página* (planos o jerárquicos) convierten esta tarea en un cometido más cómodo y sencillo.

Dada la naturaleza de las conexiones entre los numerosos componentes y la homogeneidad en importancia de cada una de las partes en que ha sido divido el diseño, es conveniente el uso de un proyecto plano en varias hojas conectadas entre sí.

3.4 Bloques y etapas

El diagrama de bloques es la representación gráfica del funcionamiento y ordenamiento internos del sistema, a través de bloques y las relaciones entre ellos. Dada la complejidad del sistema, se ha dividido en módulos que realizan etapas específicas:

- Conversor *Ethernet-RS485*, encargado de transformar la comunicación *Ethernet* a *RS-485* (transmisión diferencial) y viceversa. Consultar el anexo A.2.1.2 para conocer más acerca de estos estándares.
- Conversor *RS485-RS232*, que desempeña la función de convertir el *RS-485* a *RS-232* o a la inversa. La información pertinente sobre el estándar *RS-232* se encuentra en el anexo A.2.1.1.
- *Microcontrolador*, ocupado de procesar la información en estándar *RS-232* y realizar las acciones oportunas sobre los módulos de relés.
- *Drivers*, responsable de proporcionar suficiente corriente a las bobinas de los relés seleccionados por el *microcontrolador* para permitir su conmutación.
- Relés, que asume la misión de organizar los relés de tal manera que cumplan los requisitos de conexionado.
- Alimentación, cumple la tarea de suministrar energía a todos los elementos del sistema para su correcto funcionamiento.

Debido a la enorme cantidad de interconexiones entre los distintos bloques solo se han detallado en el diagrama las más importantes, de manera que facilite su comprensión.

La comunicación entre el ordenador y el microprocesador, mediante el paso de *Ethernet* a *RS-485* y a su vez a *RS-232* (o al contrario), otorga la posibilidad de conmutar determinados relés y establecer diferentes caminos para las señales que atraviesan los conectores.

Dado que hay una gran semejanza funcional entre módulos de relés, el hecho de que compartan el diseño de bloques supone un gran ahorro en el tiempo de desarrollo, por lo que es idóneo aprovechar cualquier parte en común de manera generalizada, de forma que sirva a todos los módulos y permita una implementación posterior de módulos diferentes más sencilla.

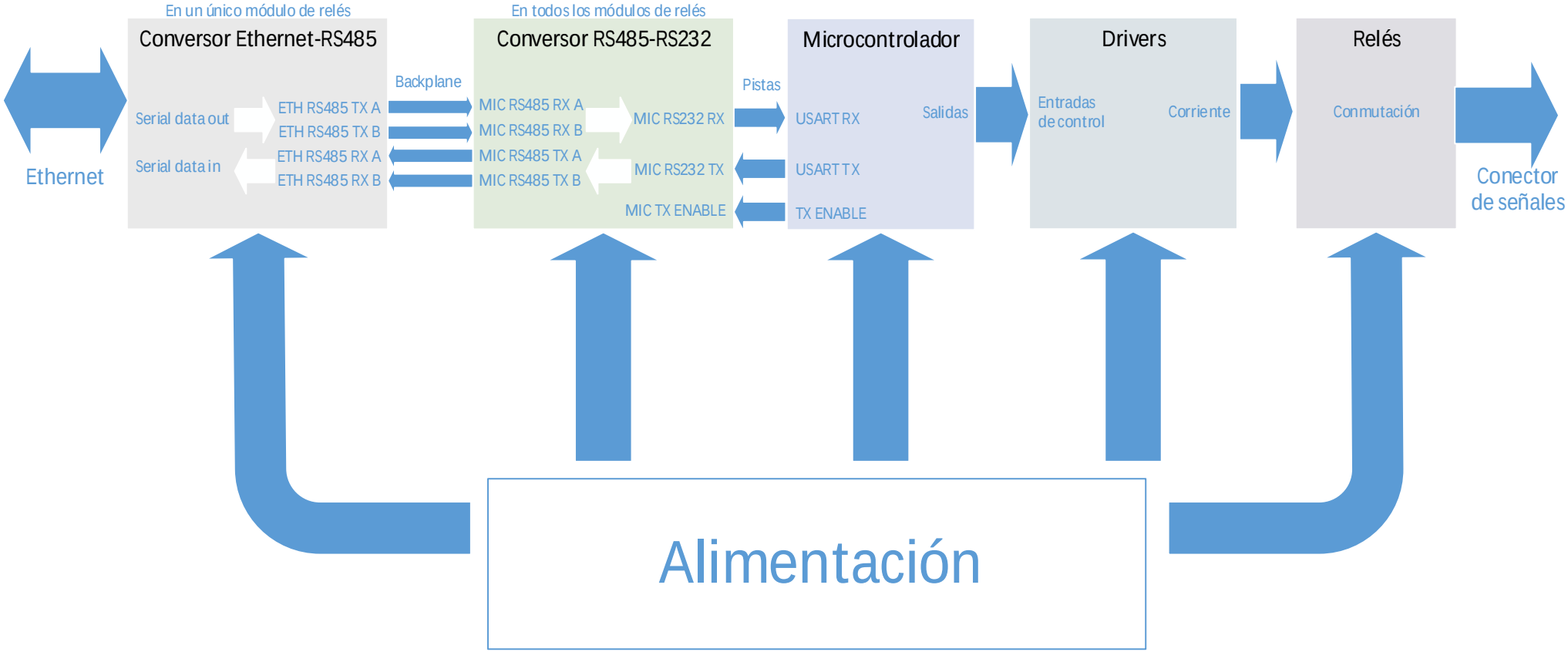


Figura 2. Diagrama de bloques

3.4.1 Etapa de Conversión *Ethernet-RS485*

El bloque Conversor *Ethernet-RS485* se ocupa de realizar esta etapa. Se parte de una conexión *Ethernet* que realiza la comunicación con el ordenador mediante un cable de estándar *Ethernet* cruzado. La conversión de *Ethernet* a estándar *RS-485* no es una tarea sencilla, por lo que se usa un conversor de *Ethernet* a *RS-232* (*XPort* de *Lantronix*); sin embargo, este protocolo, en este caso, no se ajusta estrictamente al estándar oficial de *RS-232C*, se asemeja a él y constituye un estándar no oficial (consultar anexo A.2.1.1).

Aunque pueda parecer un error convertir la comunicación de *RS-232* a *RS-485* para luego efectuar el paso inverso en otro bloque, resulta adecuado en esta aplicación porque el estándar *RS-485* garantiza una gran integridad de la señal por la transmisión diferencial y ofrece una conexión multipunto (hasta 32 conexiones). Ver anexo A.2.1.2.

La interfaz, además de constituir el medio necesario para la emisión y recepción de datos desde *Ethernet*, incorpora más características programables en sus terminales que maximizan su rendimiento:

- CP0, de cuyas funcionalidades solo es de interés el control de flujo (*RTS*).
- CP1, que carece de atractivo para este proyecto en los distintos usos que ofrece.
- CP2, con posible aplicación de provecho en este sistema como control de flujo (*CTS*).
- *Reset*, un pin de entrada que permite reiniciar el dispositivo *XPort*.

En el sistema no se utiliza control de flujo alguno por parte de estos dos pines (*CTS* y *RTS*), hay un abierto, pero se describen con mayor detalle en el anexo A.2.1.1.1.

La información se transporta en *RS-232* en dos canales (*Serial data in* y *Serial data out*) desde la interfaz *Ethernet*-Serie hasta unos *transceivers* (*ADM1485*) que se encargan de modificarla en *RS-485*, y viceversa. Para asegurar una mayor entereza de la señal, se emplean *pull ups* y *pull downs* (dependiendo del estado lógico natural del terminal), que garantizan los niveles lógicos esperados especialmente en estado de desconexión o alta impedancia, y una resistencia para evitar reflexiones de la señal (revisar anexo A.2.1.2.1).

Una vez que la comunicación transcurre en *RS-485* se puede establecer una transferencia de datos con cualquiera de los módulos; sin embargo, no se contempla el *hardware* necesario (*mainframe* o *backplane*) para su consecución.

A continuación se adjunta la figura que muestra los componentes necesarios y sus relaciones para la consecución de esta etapa.

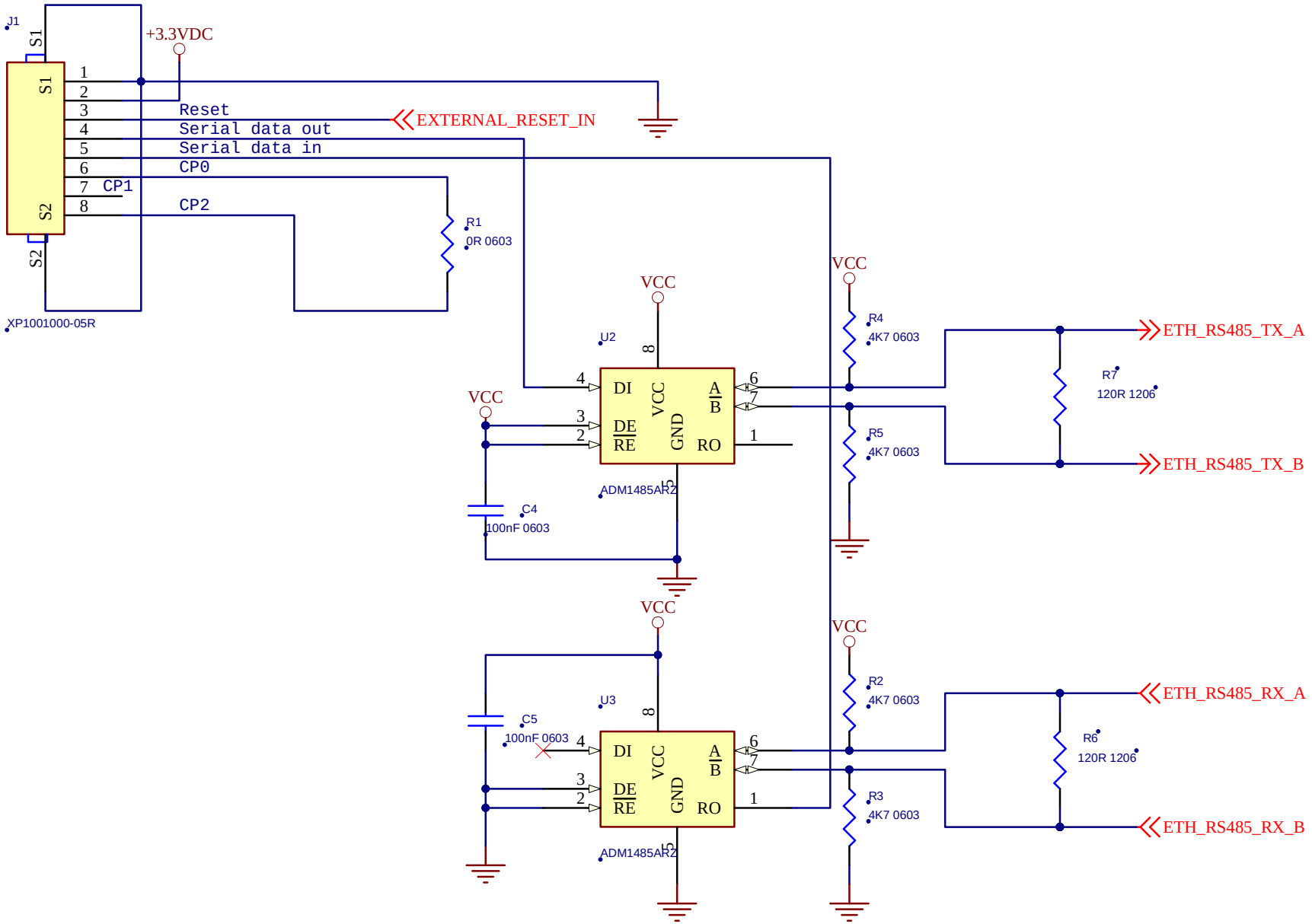


Figura 3. Esquemático de la etapa de Conversión *Ethernet-RS485*

3.4.2 Etapa de Conversión *RS485-RS232*

El módulo conversor *RS485-RS232* supone una agrupación de elementos existentes en todos los módulos, que logran la conversión del *RS-485* de la etapa anterior a *RS-232* para su procesamiento, y al contrario.

Funcionalmente constituye el cometido inverso a parte de la etapa de conversión *Ethernet-RS485*¸ la transformación del *RS-232* a *RS-485*, por lo que dada la naturaleza de los dispositivos usados anteriormente se puede conseguir este propósito con una disposición opuesta de los elementos. Sin embargo, en este paso es idónea la habilitación de los módulos (internamente el *microcontrolador*) para comunicarse con el exterior (*PC*) a través de la conversión a *RS232-RS485–RS232-Ethernet*, para lo que se utiliza una característica de los *transceivers* (*ENABLE*) que permite la transmisión solo cuando se desea, evitando que las interferencias y el ruido puedan ocasionar falsas tramas de datos y no exista posibilidad de colisión entre módulos.

Hay que tener en cuenta que como todos los módulos de relés poseen esta etapa, las reflexiones de la señal se evitan colocando resistencias al principio y final de la línea, por lo que solo un módulo de relés deberá contener dicho componente en esta etapa, aunque las resistencias en serie en las líneas (A y B) mejoran la integridad de la señal en todos los módulos. (ver anexo A.2.1.2.1).

Así mismo, como complementariedad, es posible garantizar aún mejor una buena señal de transferencia de datos en el *RS-485* incorporando factores que eliminen el ruido de alta frecuencia, como *beads,* que reaccionan con mayor resistencia ante las frecuencias altas y suponen un cortocircuito en frecuencias muy bajas.

Puesto que el sistema podría estar sujeto a operaciones con altos voltajes y corrientes, es conveniente emplear protecciones ante los posibles transitorios en las líneas de datos, ya que de otro modo un transitorio elevado podría inutilizar numerosos componentes de los módulos de relés de golpe. Los diodos de protección *ESD* aseguran la defensa frente a transitorios de voltaje elevados, preservando así la integridad de la circuitería.

A continuación se muestra la figura esquemática correspondiente que aclara todos estos detalles.

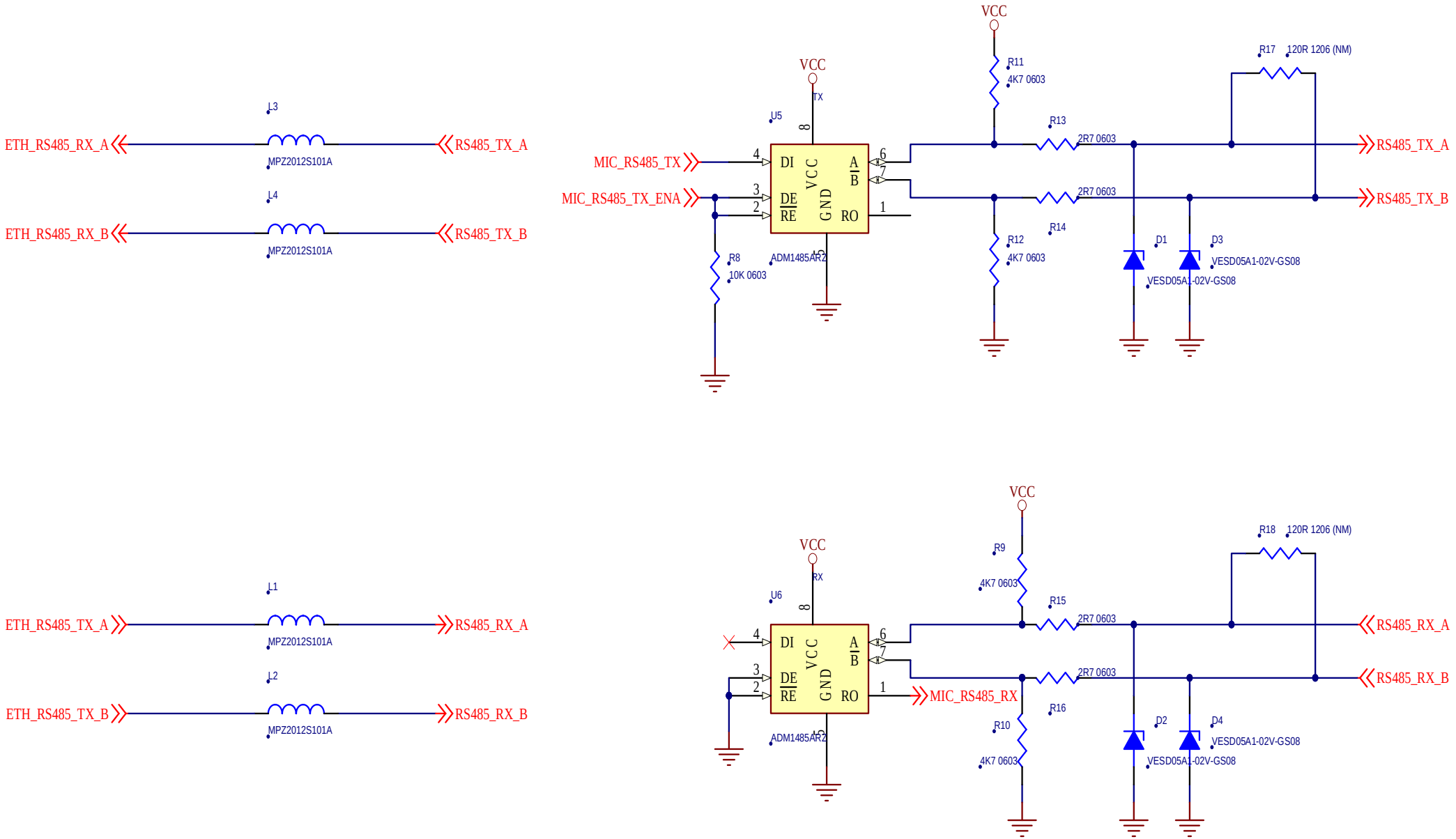


Figura 4. Esquemático de la etapa de conversión *RS485-RS232*

3.4.3 Etapa de Tratamiento de datos

La etapa de tratamiento de datos es un tramo totalmente gobernado por el *microcontrolador*, un circuito integrado programable capaz de ejecutar el código almacenado en su memoria (ver anexo A.3.1).

El código constituye el conjunto de instrucciones que acatará el *microcontrolador*, por lo que supone un elemento crítico del sistema. En procesos industrializados, con un gran volumen de unidades, la memoria de los dispositivos es grabada por el fabricante en la memoria, pero en este caso es conveniente una herramienta que permita la modificación del código para realizar ensayos y depurar el código. Por ello, en los prototipos y en series de escaso volumen de productos, para introducir esta serie de órdenes en la memoria es imprescindible una interfaz que comunique al *PC* con el *microcontrolador*, para lo que se emplean un grupo de entradas y salidas (*PROG_BOOT, PROG_TX, PROG_RST, PROG_RX, PROG_ SWCLK* y *PROG_SWIO*), así como un conector (*Molex 15-91-2100*) para posteriormente usar un adaptador hasta el ordenador.

El *microcontrolador* se comunica con el módulo conversor *RS485-RS232*, de manera que le transmite cuando debe emitir (*MIC_RS485_TX_ENABLE*), la información que debe transferir (*MIC_RS485_TX*) y recibe los datos pertinentes (*MIC_RS485_RX*). Esto evita colisiones de datos en la línea, ya que el *microcontrolador* solo transmitirá información cuando desde el PC se han referido a él. Además, el *microcontrolador* puede ordenar el reinicio del conversor de *Ethernet* por medio de la línea *EXTERNAL_RESET_IN*¸ si así lo requiere.

Para distinguir la identidad de cada uno de los módulos, de manera que solo uno de ellos responda a la llamada realizada desde el ordenador, se implementan dos formas de direccionamiento:

- Líneas de datos (*DIR1, DIR2, DIR3, DIR4* y *DIR5*) hasta un conector posterior (*DIN-048CPC-SR1*) situado en la parte trasera de la tarjeta. Así, al introducir el módulo en un *backplane* o *mainframe* dedicado (cada una de estas líneas se encuentra cortocircuitada a *+3,3VDC* o *GND*, configurando cada ranura con una dirección distinta) se *autoidentifica* con una dirección propia de la posición que ocupa.
- Un *switch* codificado (*FR01FR16H-06XL* o *S1011A*) de hasta 16 posiciones (4 contactos), que cumple de sobra con los requerimientos actuales; sin embargo, para posteriores ampliaciones existe un abierto que ejerce la función de quinto bit, por lo que se podrían utilizar hasta 32 módulos de relés a través de la misma interfaz de *Ethernet*. Este modo de proceder está orientado a la prueba de los prototipos y a la integración de los distintos módulos en una tarjeta *VXI*. Puesto que una conmutación del interruptor directa a la alimentación en alguno de sus pines supone un exceso de corriente para las entradas del *microcontrolador*, se usan resistencias de forma que limiten el paso de la corriente. [27]

Así mismo, para evitar grabar distintos códigos o *firmwares* en las memorias de los *microcontroladores* de los diferentes tipos de módulos, es conveniente la implantación de algún tipo de mecanismo que permita diferenciar una clase de módulo de otra *(IND2002A,*

IND5002 o *IND5003*). Por ejemplo, líneas que sean conectadas a *+3,3VDC* (con una resistencia en serie que limite la corriente) o *GND* y conformen una serie de bits que posibiliten discernir un modelo de módulo de otro, en este caso se han empleado 4 líneas (*CFG_SW1, CFG_SW2, CFG_SW3* y *CFG_SW4*), por lo que podrían coexistir hasta 16 tipos de módulos distintos.

La mayoría de dispositivos electrónicos poseen indicadores que señalan el estado del instrumento, lo que supone una ayuda en caso de averías o resolución de problemas. En el sistema se han incluido tres *LEDs* y sus correspondientes resistencias que adecuan la corriente que pasa por ellos, para representar los estados de encendido (*LED* verde), comunicación (*LED* amarillo), y error (*LED* rojo), por medio de salidas del *microcontrolador*.

Cualquier *microcontrolador* necesita de un circuito que le indique la velocidad de trabajo, el oscilador de frecuencia. Este circuito está constituido por un cristal y dos condensadores, que generan pulsos en los terminales (*OSC_IN* y *OSC_OUT*) del *microcontrolador*. La disposición y valores de estos elementos están determinados por el modelo de controlador (hoja de características). [36]

El modelo de *microcontrolador* empleado en el sistema ofrece numerosas salidas que cubren los requisitos de las tarjetas de relés (hasta 52 relés controlados por 52 salidas), por lo que la configuración *microcontrolador* supone una parte común para todos los módulos de relés. Estas salidas se agrupan en puertos que precisan de una alimentación (*+3,3VDC*) y masa (*GND*) para su correcto funcionamiento.

Las salidas del *microcontrolador* ofrecen un control apropiado para los relés, pero no son capaces de suministrar suficiente corriente para conseguir la conmutación de los relés, para lo que se necesita de la siguiente etapa.

La figura 5 muestra los elementos descritos que conforman la estructura y configuración del microprocesador.

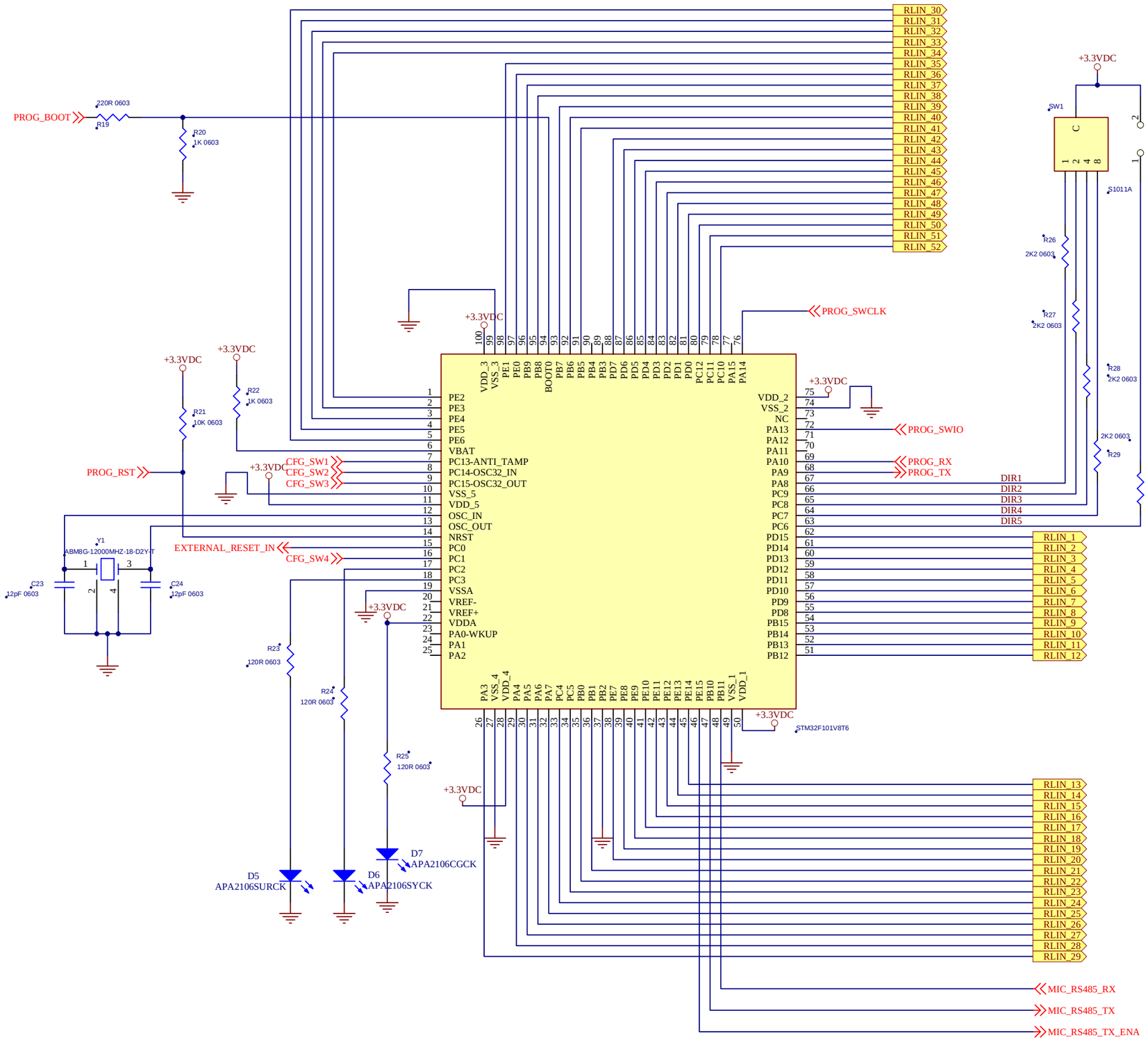


Figura 5. Esquemático de la etapa de tratamiento de datos

3.4.4 Etapa de Suministro de corriente

La etapa de suministro de corriente supone una fase que proporciona suficiente energía a las bobinas de los relés indicadas por las salidas del *microcontrolador* (un 1 lógico resulta en la conmutación del relé).

La corriente necesaria para conmutar cada relé se señala en la hoja de características (por la *ley de Ohm,* la tensión a la que están sometidos entre la resistencia que presentan), por lo que es imprescindible elegir un mecanismo que supere dicha corriente. [26]

Actualmente existen circuitos integrados dedicados a este tipo de aplicaciones, como los *drivers*, unos dispositivos electrónicos que regulan el flujo de electricidad garantizando un voltaje y corriente adecuados. En este caso se emplean *ULN2003*, siete *darlingtons* en un mismo encapsulado que proporcionan hasta 500mA por cada una de las siete salidas en colector abierto. [41]

El ULN2003 además incluye sus propias medidas de protección ante los picos de tensión provenientes de las inductancias cuando ya no circula corriente por ellas. Para ello incorpora *common free wheeling diodes* conectados a una salida *COM*, por lo que al conectarla a *VCC* (en paralelo con la carga o inductancia) el circuito integrado queda protegido. [41]

Para evitar posibles conmutaciones no deseadas debidas a interferencias, ruidos o niveles de tensión inesperados es conveniente el uso de *pull downs*, que aseguran los voltajes apropiados principalmente en desconexiones y estados de alta impedancia.

En la siguiente figura se presentan los componentes y las relaciones entre ellos recién reseñados.

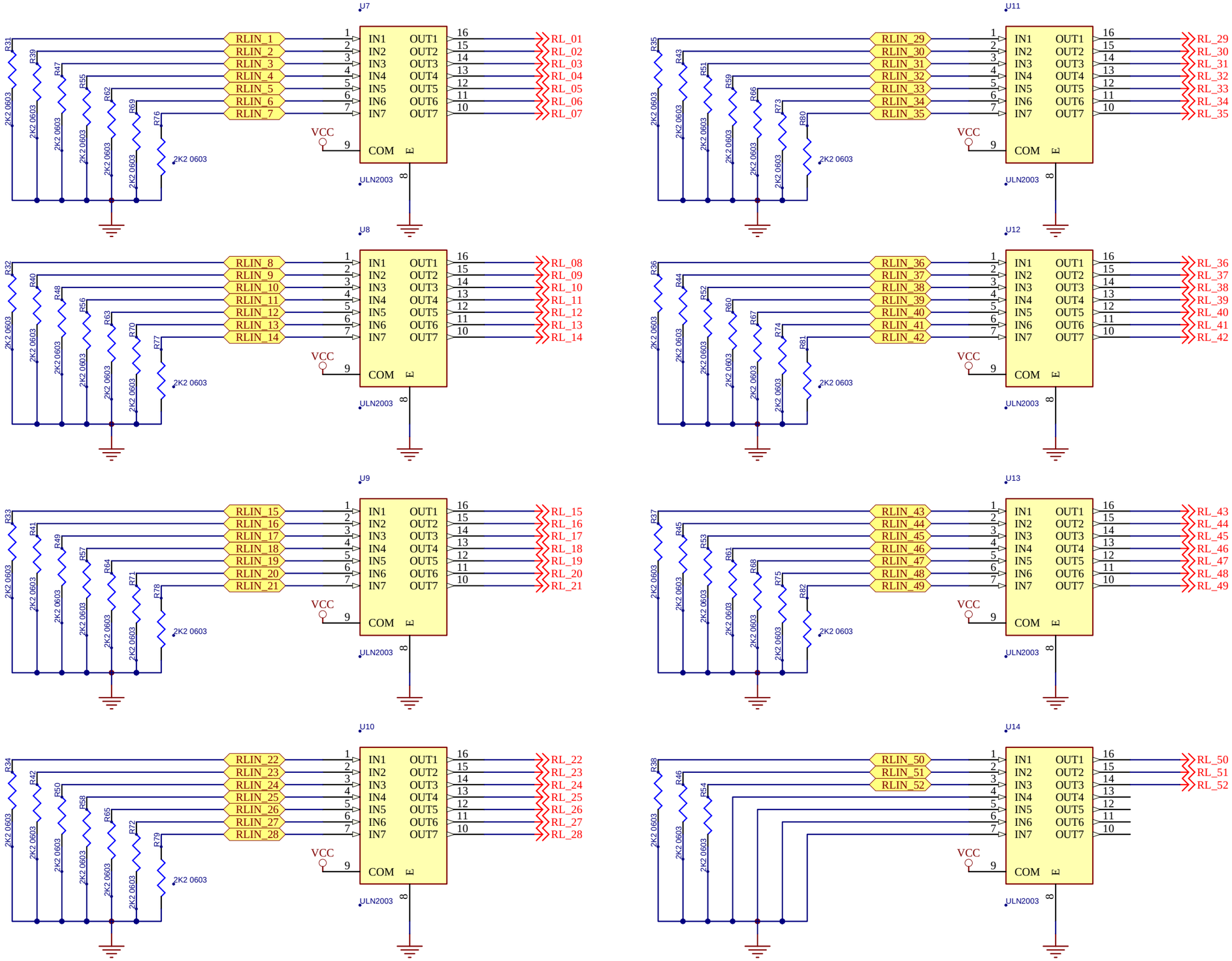


Figura 6. Esquemático de la etapa de suministro de corriente

3.4.5 Etapa de Configuración de relés y características específicas

Cada módulo de relés posee un conector y una configuración de relés establecidos por los requisitos, así como una configuración *hardware* (*CFG_SW1, CFG_SW2, CFG_SW3* y *CFG_SW4*) y una etiqueta (*LABEL*) distinta, que los diferencia.

3.4.5.1 Módulo IND2002A

El módulo de relés *IND2002A* debe estar constituido por relés *AZ944-1C-5DE* dispuestos en una organización *SPDT*; sin embargo, aunque todavía continúan en el mercado comienzan a estar obsoletos, por lo que es imperativo buscar una solución. El relé *RTD14005F* presenta características eléctricas superiores, aunque implica la ocupación de más espacio en el módulo y conlleva adaptaciones en el diseño del hardware a considerar en etapas de desarrollo posteriores. Dado que no requiere de una configuración *DPDT*, la alternativa más adecuada consiste en cortocircuitar las posiciones semejantes entre ellas, de modo que funcione como un relé con un contacto común, un contacto normalmente abierto y un contacto normalmente cerrado, de manera que cada relé *RTD14005F* permita estar sometido a más corriente y adopte la estructura adecuada, como señala la figura 7. [45]

Figura 7. Relé *SPDT*

La configuración de relés y su conexión con el conector correspondiente supone una tarea sencilla mediante el uso de *Altium* (consultar *plano IND2002 - 1* en anexo A.5.2).

3.4.5.2 Módulo IND5002

El módulo de relés *IND5002* está comprometido a emplear relés *EC2-5NJ* en una configuración *SPDT* y dado que cada uno de estos relés está formado en su interior por dos varillas diferentes que conmutan al mismo tiempo dando lugar a dos posiciones comunes, dos posiciones normalmente abiertas y dos posiciones normalmente cerradas, todas independientes pero gobernadas por la misma bobina. De nuevo, la relación entre los relés y el conector del módulo no resulta complicada esquemáticamente (ver planos *IND5002 - 1* e *IND5002 – 2* en A.5.2) a través de herramientas dedicadas. [26]

Los relés *EC2-5NJ* poseen una configuración *DPDT*, así que es provechoso realizar una conexión como en los relés del módulo *SMP2002A*, cortocircuitando las posiciones similares y aumentando la capacidad al paso de corriente.

3.4.5.3 Módulo IND5003

El módulo de relés *IND5003* está obligado a presentar una estructura *SPQT* con relés *EC2-5NJ*, es decir una sola posición común y cuatro posibles posiciones de conmutación. Puesto que el relé *EC2-5NJ* dispone de posiciones independientes es posible conseguir una solución con tan solo dos relés. [26]

El primer relé se conecta cortocircuitando las posiciones análogas, como en el módulo *IND5002*, y las posiciones normalmente abierta (*NO*) y normalmente cerrada (*NC*) se conectan a las posiciones comunes (*COM*) independientes del siguiente relé, de forma que el segundo relé ofrece los cuatro posibles caminos del conjunto *SPQT* y el primer relé otorga la posición común. La siguiente figura esclarece la conexión entre dos relés.

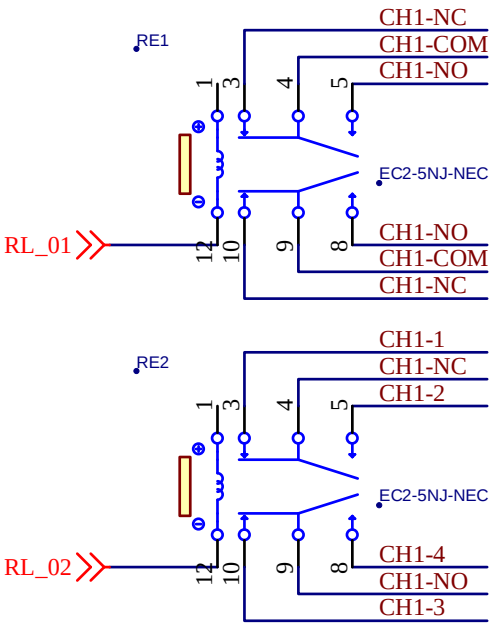


Figura 8. Conexión de los relés del módulo *IND5003*

Así, agrupados en parejas, es posible controlar la conmutación a cuatro posiciones mediante dos señales (dos bits) y funcionalmente operar como un único relé como muestra la figura 9.

Figura 9. Relé *SPQT*

Este conexionado tiene el inconveniente de que la señal proveniente de la posición común puedan llegar hasta posiciones indeseadas, en los supuestos de querer trasladar una señal que se encuentra aplicada en la posición 1 a la posición 4, o de la posición 2 a la posición 3, requiere la conmutación de ambos relés, pero si uno de los relé conmuta antes que el otro resultaría en la aparición de la señal aplicada en la posición común a una posición no deseada. Por ejemplo, en el caso de dirigir una señal desde la posición 1 a la posición 4 y el primer relé, al que le es suministrado directamente la señal en cuestión, conmutase primero, aparecería la señal momentáneamente en la posición 3, y a continuación tras la conmutación del segundo relé se hallaría correctamente en la posición 4. Este problema es resoluble mediante el *software* si atendemos a que en las hojas de características de los relés la varilla tarda en moverse de un contacto a otro (*APPLIED VOLTAGE VS. TIMING*) lapsos de la escala de milisegundos (2ms aproximadamente). El *microcontrolador* trabajará a una frecuencia de MHz, por lo que con establecer las ordenes de cerrar un relé y a continuación el segundo, sin ninguna espera, el intervalo de tiempo entre las aplicaciones de tensión en respectivas salidas del *microcontrolador* será del orden de microsegundos, despreciables frente al tiempo que tarda en conmutar el relé. [26]

La conexión entre las posiciones y el conector es una labor elemental con los programas *CAD* (consultar ver planos *IND5003 - 1* e *IND5003 – 2* en A.5.2).

3.4.6 Etapa de Alimentación

La etapa de alimentación proporciona la energía necesaria a la circuitería del módulo de relés; sin embargo, dado que *VXI* o un posible *backplane* suministra *+5V* (*VCC*) estables, solo es necesaria la implementación de la alimentación de *+3.3VDC*. Un regulador lineal mantiene constante una tensión de salida, de forma que garantiza *+3.3VDC*. Hay que observar detenidamente el consumo del circuito para elegir apropiadamente un modelo u otro de regulador, en el sistema, dado que cada módulo de relés posee su propio bloque de alimentación y en el peor de los casos cuenta con el bloque conversor *Ethernet-RS485* no es necesario un gran amperaje en la salida para el correcto funcionamiento del sistema. El elemento que más consume es la interfaz *Ethernet-Serie*, ya que tal y como indican las hojas de características, dependiendo del tipo de *Ethernet* puede llegar a necesitar hasta 233mA, lo que convierte el consumo del resto de elementos en un valor casi despreciable. Teniendo en cuenta que es necesario un margen de seguridad y redondeando muy al alza el gasto energético de la circuitería alimentada a *+3.3VDC* (unos 300mA), cualquier regulador de *+5V* a *+3.3VDC* por encima de 600mA es un candidato adecuado.

El regulador *LM1117-3.3* ofrece hasta 800mA de corriente en su salida de *+3.3VDC*, por lo que se ajusta a las necesidades del sistema. [43]

En los sistemas de alimentación es conveniente la implantación de condensadores en las cercanías de cualquier elemento conectada a dicha alimentación para eliminar cualquier tipo

de oscilación o rizado, por lo que se emplean condensadores de 100nF en cada alimentación, tanto de *+5V* (*relés*, *drivers, transceivers* y regulador) como de *+3.3VDC*, y en el caso del regulador condensadores de mayor capacitancia en paralelo. Los condensadores de mayor valor (tantalio) eliminan con mayor eficacia los rizados más elevados; sin embargo, responden peor que los de baja capacitancia (cerámicos) a los cambios bruscos de tensión. La siguiente figura muestra el esquema de conexionado del regulador y los condensadores.

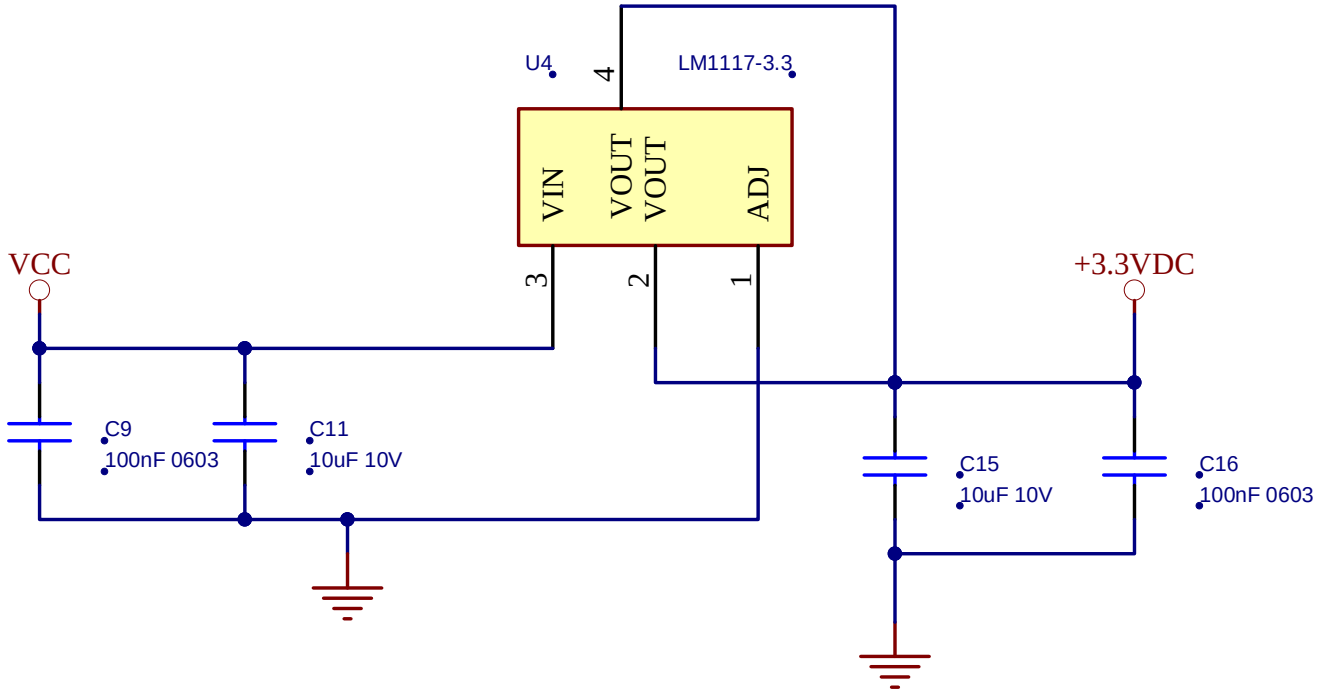


Figura 10. Esquemático del regulador del circuito

3.4.7 Conectores generales

Los conectores comunican el sistema (cada módulo) con el exterior, por lo que para constituir este vínculo es adecuado elegir conectores estandarizados y que cumplan con los requisitos y procuren una fácil expansibilidad del sistema:

- Un conector (*Phoenix 1755794*) que enlaza con la alimentación los elementos del circuito dedicado especialmente para las pruebas y depuración de los prototipos (*VCC, +12V, +24V, -5.2V -12V -24V* y *GND*). Aunque no es necesario actualmente para cualquiera de los módulos de relés, es conveniente establecer un medio para otras posibles alimentaciones que puedan usarse en un futuro sin muchas modificaciones en el *hardware*. [28]
- Un conector (*Molex 15-91-2100*) que permite la inserción de un adaptador de grabación de *microcontroladores* con las líneas (*PROG_BOOT, PROG_TX, PROG_RST, PROG_RX, PROG_ SWCLK* y *PROG_SWIO)*. [18]

- Un conector trasero (*DIN-048CPC-SR1*) que conforma el canal de comunicación del estándar *RS-485* (*ETH_RS485_TX_A, ETH_RS485_TX_B, ETH_RS485_RX_A y ETH_RS485_TX_B*), establece el nexo de la configuración de direcciones entre el *backplane* o *mainframe* y las líneas del *microcontrolador* (*DIR1, DIR2, DIR3, DIR4* y *DIR5*), y conexiona con la alimentación de la circuitería (*VCC, +12V, +24V, -5.2V -12V -24V* y *GND*) proveniente del *backplane*. [5]
- Un conector (*FCI 74543-3661LF*) principalmente dedicado a la conexión entre prototipos y entre tarjetas compuestas por los módulos de relés, que desempeñan la función de comunicar en estándar *RS-485* (*ETH_RS485_TX_A, ETH_RS485_TX_B, ETH_RS485_RX_A y ETH_RS485_TX_B*) las distintas tarjetas o instrumentos que contienen una única interfaz *Ethernet-Serie*.

3.4.8 Resumen

El bloque conversor *Ethernet-Serie* establece el medio de comunicación entre el ordenador (por *Ethernet*) y los diversos módulos de relés, a través de una interfaz realiza una transformación *Ethernet-RS232* para a continuación efectuar una conversión *RS232-RS485*, o viceversa, de manera que un único módulo que contenga este bloque permite gobernar hasta 31 módulos más a través del estándar *RS-485*. Así mismo este bloque permite ser separado de los módulos, de forma que exista en un posible *mainframe* al que se conecten los módulos.

Cada módulo de relés del sistema posee un bloque conversor *RS485-RS232* que permite que la información proveniente del bloque conversor *Ethernet-Serie* sea interpretado por el *microcontrolador*. Este a su vez, analiza si la trasmisión se refiere al módulo en el que se encuentra, procesa la información y otorga una respuesta y las acciones pertinentes si las hubiese. Entonces la información sigue el camino inverso hasta el ordenador.

En caso de que la comunicación entre ordenador y módulo le indique a este último que necesita modificar los caminos de algún relé, el *microcontrolador* del módulo habilita las salidas adecuadas, teniendo en cuenta que cada módulo posee una disposición de relés diferente, que controlan los *drivers* apropiados, y estos suministran a los relés la corriente necesaria para su conmutación.

3.5 Diseño de la *PCB*

Esta fase del desarrollo del hardware exige una completa elaboración de los documentos esquemáticos de todos los módulos de relés, ya que consiste en la ubicación de todos los componentes en la tarjeta (módulo de relés) y en su conexionado por medio de pistas. Los planos esquemáticos suponen la base de todo el diseño hardware de la tarjeta de circuito impreso, por lo que hay que prestar especial atención a la conexión entre componentes, además de evitar que pines sin conectar actúen como antenas (cortocircuitándolos a *GND* o la

alimentación dependiendo del caso), interfiriendo en el funcionamiento correcto del dispositivo. Para ello *Altium Designer* incluye un compilador que detecta automáticamente posibles errores de conexión tras haber completado los plano esquemáticos, facilitando la labor del desarrollador. El análisis del resultado de la compilación permite depurar el esquemático; sin embargo, si aún existiesen errores es posible modificar el esquemático y aplicar los cambios en la etapa de diseño de la *PCB* (*Printed Circuit Board*).

El proyecto creado en *Altium Designer* permite compartir documentos comunes, así como incorporar nuevos documentos. El modelado de la tarjeta de circuito impreso es una tarea que debe llevarse a cabo en un documento específico dentro del proyecto, de manera que las transformaciones en cualquiera de los documentos pertenecientes al proyecto se actualicen en el resto. Una vez creado el documento de *PCB*, se actualizan todos los elementos y sus relaciones desde los planos esquemáticos.

Altium Designer incorpora multitud de *footprints* (huellas) que pueden representar los elementos de los esquemáticos en el proceso de diseño de la tarjeta de circuito impreso, aunque en caso de no encontrar una huella adecuada es posible elaborarla a medida en la *librería PCB*.

El tamaño de la tarjeta está delimitado por los requisitos técnicos (un máximo de 11 cm de alto hasta 34 cm de profundidad y un límite de ancho de 2 cm), es por ello que es conveniente partir de estos límites y una vez que todos los elementos del sistema están correctamente ubicados y con un espaciado adecuado, reducir el tamaño de la tarjeta de circuito impreso hasta que el diseño lo permita, lo que abarata ligeramente el precio de cada módulo.

Dado que los módulos de relés comparten muchos de los bloques en común (conversor *Ethernet-RS485*, conversor *RS485-RS232*, *Microcontrolador* y *Drivers*), diseñar dicha parte diferenciable de manera colectiva supone un ahorro significativo en el tiempo de desarrollo del documento de la tarjeta de circuito impreso:

1. En primer lugar es imprescindible situar los componentes del módulo de relés de manera que reduzcan la complejidad del rutado posterior de pistas, por lo general siguiendo los esquemáticos de un lado hacia otro. También hay sopesar la situación de los conectores, aunque algunos de ellos se encuentren condicionados (en la parte trasera *DIN-048CPC-SR1* y en la parte delantera, a la vista, *GMCT41M* y *DIN 41612 004778*), es conveniente situar los conectores poco utilizables en partes poco accesibles, y viceversa. Dadas las dimensiones de los módulos, situar la interfaz *Ethernet-Serie* (*Xport*) en los módulos *IND5002* e *IND5003* es físicamente imposible debido a que el conector *DIN 41612 004778* ocupa la mayor parte de la altura establecida (máximo 11 cm) [9], es por ello que se ubican en la parte posterior, donde gozarán de gran utilidad en las pruebas con los prototipos y sin embargo, raramente se implanten en el sistema final, ya que solo es necesario una interfaz *Ethernet-Serie* para los módulos y se insertará preferiblemente en el módulo *IND2002A*. A continuación se presenta la estructuración de los elementos comunes en la figura 11.

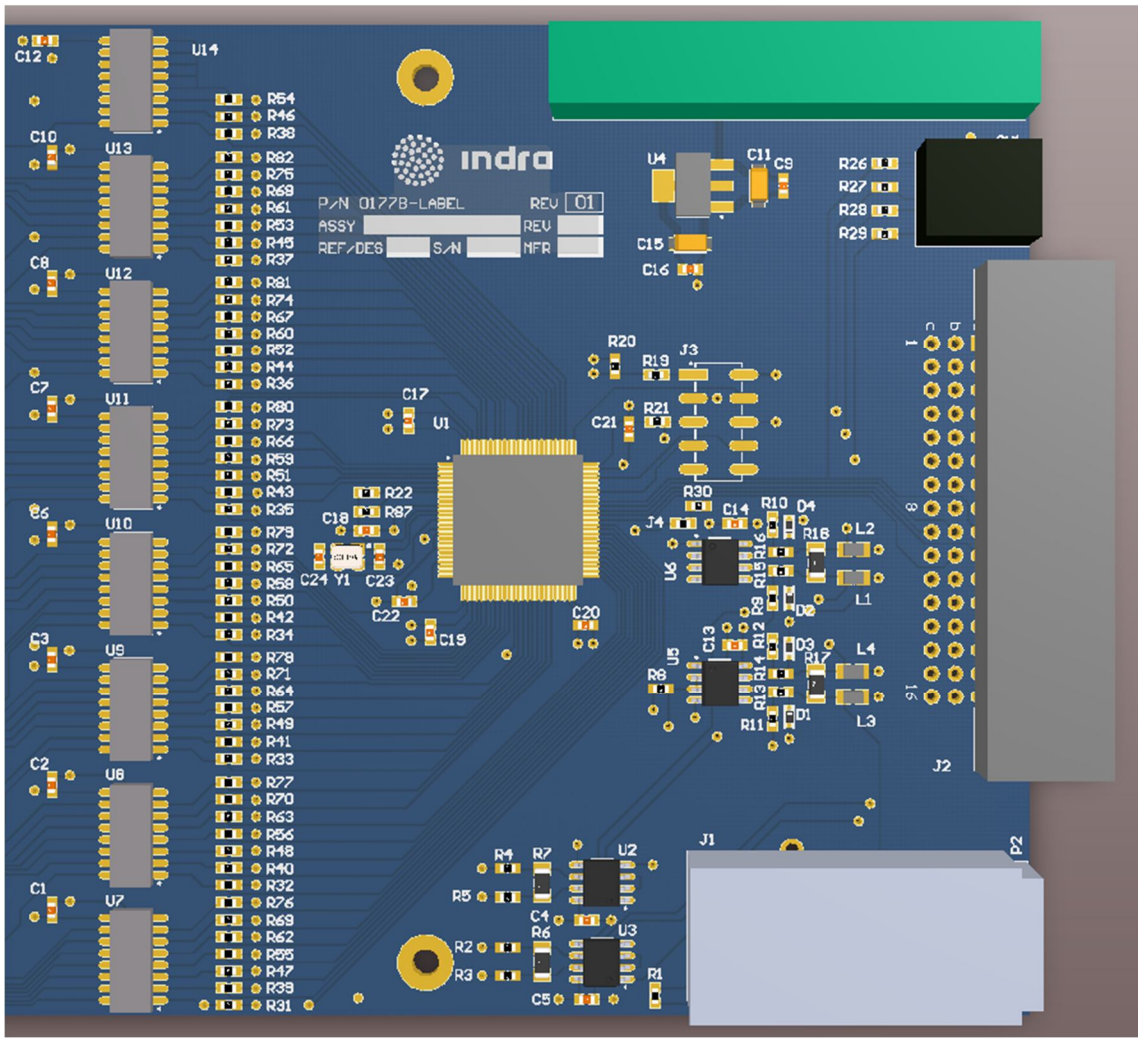


Figura 11. Diseño *hardware* común de los módulos *IND5002* e *IND5003*

Tal y como se observa en la figura 11, que representa una vista frontal (tridimensional) de los componentes en la *PCB*, se sitúan los elementos del módulo siguiendo un orden lógico establecido por los esquemáticos del sistema y con cierta proximidad a los bloques con los que guardan alguna relación. Los condensadores se localizan cerca de cualquier pin de componente que esté conectado a la alimentación.

En cuanto a los conectores, de arriba abajo se sitúan el conector verde de alimentación (*Phoenix 1755794)*, el *switch* negro que permite configurar la dirección del módulo, el conector trasero *DIN-048CPC-SR1*, y en el caso de los módulos *IND5002* e *IND5003* la interfaz *Ethernet-Serie*.

También se han introducido los elementos comprendidos por todos los módulos, como marcadores *fiducial* (usados como punto de referencia para el posicionamiento físico automatizado de los componentes) y taladros (*Drills*) que facilitan su acoplamiento a otros elementos posteriormente.

2. Una vez que los componentes han sido colocados conformando una estructura lo más simple posible, se procede a formalizar físicamente las conexiones entre los componentes, es decir, el rutado. El rutado puede establecerse automáticamente o manualmente; sin embargo, un rutado automático organiza las pistas de manera

arbitraria, es por ello que es una alternativa dedicada a circuitos muy simples, como no es el caso. El rutado manual consiste en colocar segmentos de pistas hasta conectar completamente dos elementos.

La complejidad del rutado, influido por las relaciones de los elementos y la anchura de pista, determina el número de capas necesarias para lograr enlazar todos los elementos. En pistas comunes que recorren muchos elementos, como las alimentaciones y tierra, y en aislamientos, es muy eficaz emplear polígonos conductores (*polygon pour*) para asegurar conductividad en una zona completa por medio de vías, es por ello que las alimentaciones (*VCC* y *+3.3VDC*) garantizan el suministro de energía en forma de polígonos apropiadamente divididos con mayor capacidad de corriente que cualquier pista, en una sola capa (*Mid-Layer-1*). La masa, común para todos los componentes, se constituye de igual manera en otra capa (*Mid-Layer-2*).

Aunque el rutado sea manual, *Altium Designer* fija las conexiones según los planos esquemáticos, permite la creación de reglas (espacio libre, anchuras, etc.) para personalizar las pistas y polígonos de la *PCB*, y ofrece herramientas para la elaboración de cualquier rutado. El ancho de pista constituye un factor a tener en cuenta para las pistas recorridas por una corriente elevada, por lo que para estos casos es indispensable realizar los cálculos necesarios para determinar el ancho de pista requerido (ver anexo A.4.2).

Completado el diseño de circuito impreso de las partes comunes a todos los módulos se puede clonar en su totalidad para cada módulo de relés y realizar las adaptaciones necesarias para el cumplimiento de requisitos. De este modo, los bloques en común de los módulos *IND5002* e *IND5003* mostrarán una estructura prácticamente idéntica (cada módulo integra configuraciones distintas de las líneas *CFG_SW1, CFG_SW2, CFG_SW3* y *CFG_SW4)*, mientras que los bloques del módulo *IND2002A* presentan pequeñas diferenciaciones resultado de la eliminación parcial (algunos *drivers* y las pistas que los relacionan) o traslado de bloques funcionales (conversor *Ethernet-RS485*), tal y como muestra la figura 12.

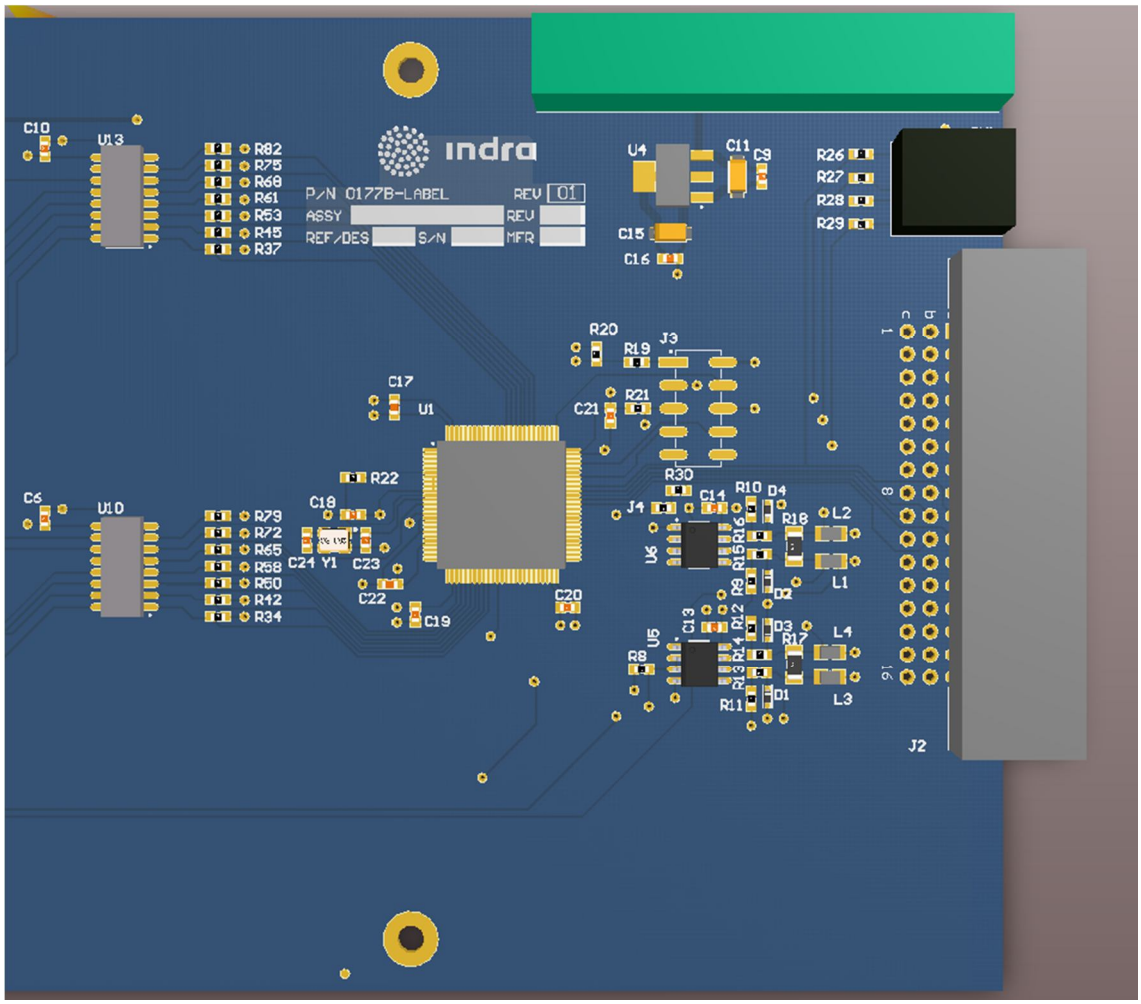


Figura 12. Diseño *hardware* común del módulo *IND2002A*

Los requisitos técnicos establecen la configuración de los relés de cada módulo, por lo que el resto de la tarjeta de circuito impreso conforma un diseño independiente para cada módulo; sin embargo, la complejidad de la disposición y rutado de los componentes restantes supone una labor más sencilla. Cabe destacar que las pistas de las señales de los relés pueden ser portadores de señales de alta tensión y/o corriente, es por ello que hay que prestar atención a la anchura y grosor de las pistas (consultar anexo A.4.2).

La dificultad del rutado, ha condicionado el grosor de las capas de cobre y su número:

- El módulo de relés IND2002A precisa de 4 capas de cobre de 35 micras cada una.
- Los módulos de relés *IND5002* e *IND5003* requieren de 6 capas de cobre de 70 micras cada una.

Además de un diseño dispar con el resto de módulos de relés, el módulo *IND2002A* presenta ciertas singularidades:

- Un conector delantero que no presenta una conectividad directa con la *PCB*, los pines del conector *GMCT41M* exigen una conexión cableada, es por ello que cerca de los relés *RTD14005F* existen taladros para todas las posiciones, de diámetro determinado

por la corriente que son capaces de aguantar (ver anexo A.4.2), donde insertar y soldar los cables. [31]

- Una anchura que podría comprometer las especificaciones (máximo de 2cm). Los relés *RTD14005F* indican en la hoja de características una altura de 15.7 mm desde la placa *PCB*, unido a un grosor de unos 3mm de la tarjeta de circuito impreso y los pines de componentes de inserción (especialmente relés), lo que resulta en 18.7 mm, muy cercanos a los 2 cm especificados. Aunque los relés actuales (altura moderada) no constituyen una amenaza contra los requisitos, es posible que si por cualquier razón hubiese que emplear otros relés más altos la *PCB* quedaría inservible. Una alternativa, apropiada dadas la circunstancias, consiste en tumbar los relés sobre la tarjeta de circuito impreso y la implementación de ranuras con terminales soldables en la *PCB* (última capa, *Bottom Layer*), de modo que una pequeña tarjeta de circuito impreso soldada a los pines del relé permita tumbar al relé mientras encaja en las ranuras y mediante una soldadura se unan los terminales de las pequeñas tarjetas de circuito impreso con los terminales de la ranura. [45]
  Así, aunque la configuración actual no lo requiere, existe una ranura por cada relé, que facilitaría la conexión de los relés de excesiva altura a la *PCB*.

De esta forma, los módulos de relés presentan la consecuente distribución de elementos en la placa expuestos en las figuras 13, 14 y 15.

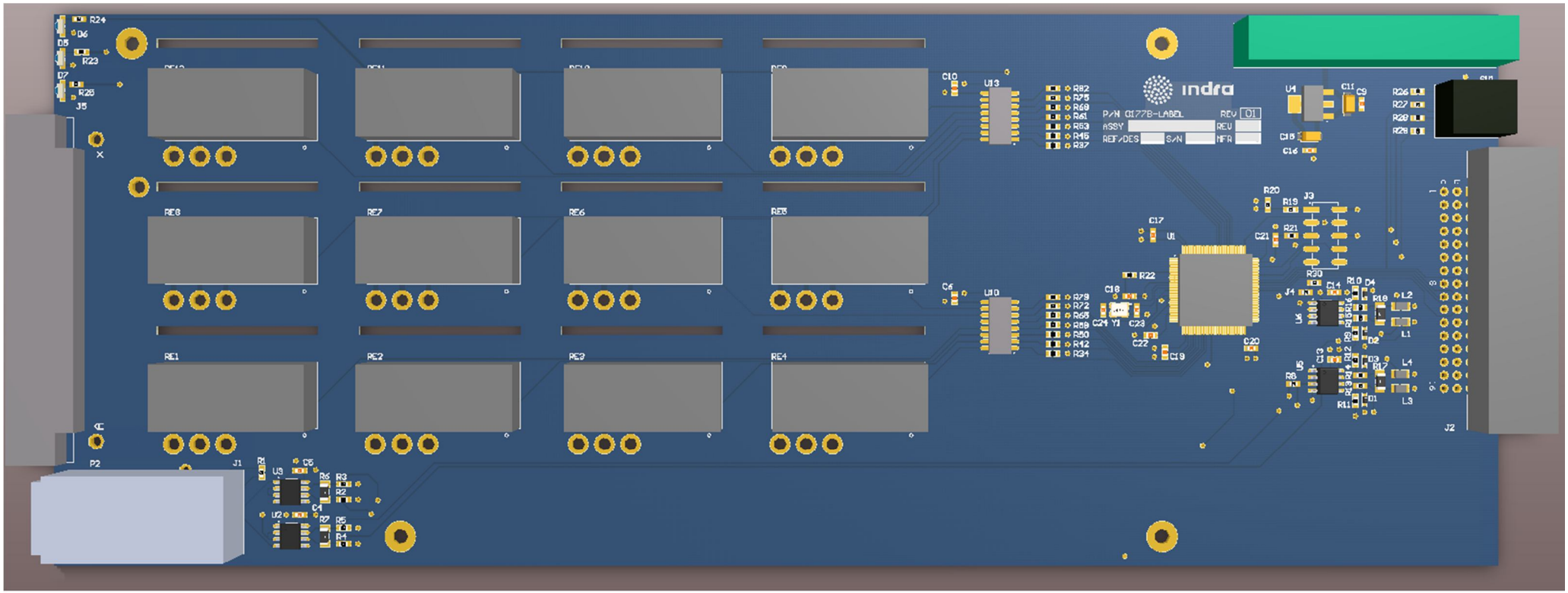


Figura 13. Vista tridimensional del módulo de relés *IND2002A*

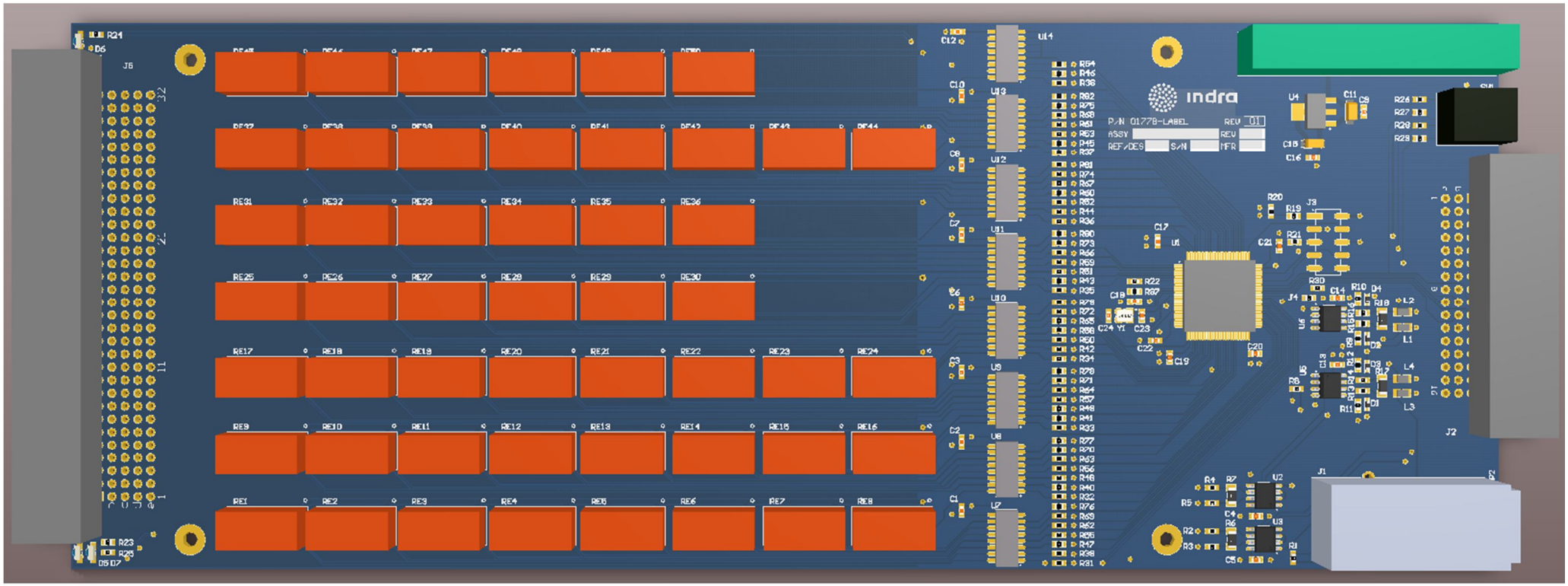


Figura 14. Vista tridimensional del módulo de relés *IND5002*

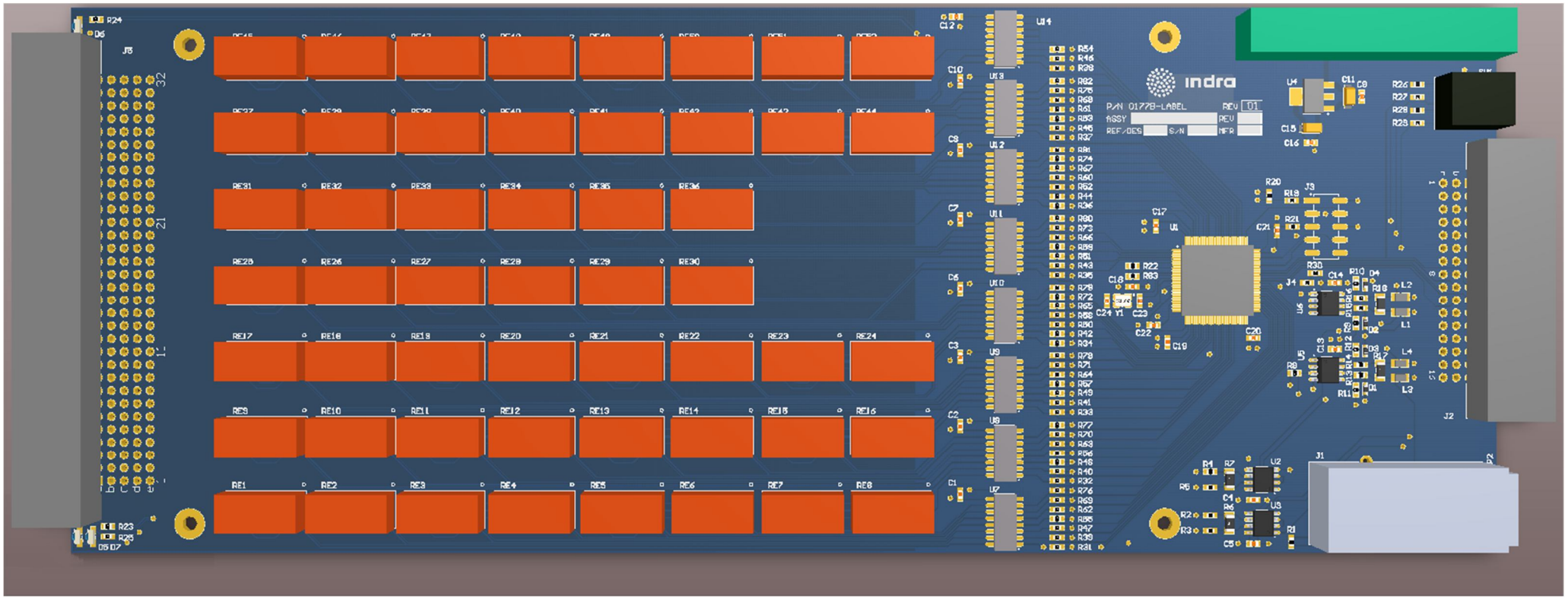


Figura 15. Vista tridimensional del módulo de relés *IND5003*

Tal y como establecen los requisitos, para garantizar una buena integridad de la señal que atraviesa los relés, es indispensable un buen aislamiento. Si bien es factible la introducción de capas adicionales de aislamiento, también lo es situar zonas de conducción (polígonos) de *SHIELD* en las capas existentes, para aislar las señales del ruido y de las interferencias. De esta forma en todas las capas hay polígonos conectados a *SHIELD*, que conforman un gran aislamiento.

Concluido el posicionamiento y el rutado entre los elementos del sistema, *Altium Designer* ofrece una herramienta de verificación de las reglas por defecto o personalizadas (*Design Rule Check*), es por ello que conviene ejecutarlo para advertir los posibles errores en el diseño de las *PCB*. Tras depurar todos fallos, se comienza con la fabricación de los módulos de relés, para lo que es necesario generar los ficheros de producción, *Gerber* y *Drills,* a través de *Altium Designer*, que indican con exactitud todos los elementos que conforman cada capa de las tarjetas de circuito impreso (ver planos de ensamblaje en anexo A.5.2). Estos ficheros, entregados a una empresa dedicada a la manufactura de tarjetas de circuito impreso, indican a la maquinaría como proceder para lograr construir el diseño desarrollado.

3.6 Materiales y componentes

El funcionamiento descrito por etapas representa una leve reseña sobre el comportamiento general de los componentes, que puede dar lugar a dudas sobre la elección del elemento frente a otras alternativas. Aunque cada componente ha sido meticulosamente calculado (revisar anexo A.4.1) solo existen algunos elementos importantes, que marcan el diseño del sistema y de cada bloque. El estudio de los principales constituyentes de cada bloque determinante ofrece una visión más completa acerca de su cometido y esclarece las opciones sometidas a juicio antes de la selección.

3.6.1 Lista de componentes

Los esquemáticos electrónicos se han conformado de tal manera que comparten numerosos componentes, lo que resuta en un uso compartido de la mayoría de esquemáticos y un ahorro significativo del tiempo de desarrollo. La lista de componentes ofrece una visión global de los componentes empleados, calculados previamente (ver anexo A.4.1).

En general, casi todos los componentes del circuito intervienen en todos los módulos de relés, aunque en una etapa posterior de desarrollo, el diseño de la *PCB*, se eliminen algunos elementos innecesarios. En la lista de componentes de la tabla 1 se describen los elementos en tres campos, los *designadores* correspondientes en el esquemático, el debido valor or *Part Number* y la cantidad de componentes bajo el mismo valor o *Part Number*. Si bien existen sutiles diferencias entre los esquemáticos de un módulo y otro, los elementos que únicamente

pertenecen a un tipo de módulo de relés se representan con asteriscos (*IND2002A *, IND5002 *** e *IND5003 ****).

Designador	Valor/ Part Number	Cantidad
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C12, C13, C14, C16, C17, C18, C19, C20, C21, C22	100nF 25V 0603	20
C11, C15	10uF 10V 1206	2
C23, C24	12pF 50V 0603	2
D1, D2, D3, D4	VESD05A1-02V-GS08	4
D5	APA2106SURCK	1
D6	APA2106SYCK	1
D7	APA2106CGCK	1
J1	XP1001000-05R	1
J2	DIN-048CPC-SR1	1
J3	Molex 15-91-2100	1
J4	2556-T02C	1
J5	GMCT41M0T0000* DIN 41612 004778 ** ***	1
L1, L2, L3, L4	MPZ2012S101A	4
LA1	LABEL	1
P1	Phoenix 1755794	1
P2	FCI 74543-3661LF	1
R1	0Ω 0603	1
R2, R3, R4, R5, R9, R10, R11, R12	4,7kΩ 0603	8
R6, R7	120Ω 1206	2
R8, R21	10kΩ 0603	2
R13, R14, R15, R16	2,7Ω 0603	4
R17, R18	120Ω (NM) 1206	2
R19	220Ω 0603	1
R20, R22, R87**, R83***	1kΩ 0603	2* 3** ***
R23, R24, R25	120Ω 0603	3
R26, R27, R28, R29, R30, R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42, R43, R44, R45, R46, R47, R48, R49, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59, R60, R61, R62, R63, R64, R65, R66, R67, R68, R69, R70, R71, R72, R73, R74, R75, R76, R77, R78, R79, R80, R81, R82	2,2kΩ 0603	57
RE1, RE2, RE3, RE4, RE5, RE6, RE7, RE8, RE9, RE10, RE11, RE12, (RE13, RE14, RE15, RE16, RE17, RE18, RE19, RE20, RE21, RE22, RE23, RE24, RE25, RE26, RE27, RE28, RE29, RE30, RE31, RE32, RE33, RE34, RE35, RE36, RE37, RE38, RE39, RE40, RE41, RE42, RE43, RE44, RE45, RE46, RE47, RE48, RE49, RE50)*** ***, (RE51,RE52)***	AZ944-1C-5DE* EC2-5NJ-NEC** ***	12* 50** 52***
SW1	S1011A	1
U1	STM32F101V8T6	1
U2, U3, U5, U6	ADM1485ARZ	4
U4	LM1117-3.3	1
U7, U8, U9, U10, U11, U12, U13, U14	ULN2003	8
Y1	ABM8G-12000MHZ-18-D2Y-T	1

Tabla 1. Lista de componentes

3.6.2 Bloque Conversor *Ethernet-RS485*

El bloque conversor *Ethernet-RS485* desempeña la labor, como su propio nombre indica, de realizar la transformación *Ethernet-RS485* fundamentalmente a través de los siguientes componentes:

- *U2*, *U3* (*ADM1485ARZ*) ➔ Es un emisor/receptor diferencial dedicado a comunicaciones multipunto de alta velocidad, *RS-485* y *RS-232* entre ellas. Tanto la emisión como la recepción se habilitan independientemente, y en caso de no hacerlo las salidas muestran un estado de alta impedancia. En el caso de *U2*, el dispositivo está destinado únicamente a la transmisión (*DE* y $\overline{RE}$ cortocircuitados a *Vcc*), la información es recibida en *DI* (*Data IN*) y se convierte al estándar *RS-485* (diferencial) por las salidas *A* (*ETH_RS485_TX_A*) y $\overline{B}$ (*ETH_RS485_TX_B*). *U3* se encuentra configurado de manera inversa, se destina a la recepción *(DE* y $\overline{RE}$ cortocircuitados a *GND*), de forma que los datos (transmitidos de manera diferencial desde el bloque conversor *RS485-RS232*) son recibidos en *A (ETH_RS485_RX_A)* y $\overline{B}$ (*ETH_RS485_RX_B)* y son transformados a *RS-232* en *RO* (*Data OUT*). [39]
  Existen multitud de *transceivers* como este en el mercado, con huellas compatibles; sin embargo, este modelo proporciona una velocidad hasta 30 Mbps, una comunicación extremadamente rápida.
- *J1* (*XP1001000-05R*) ➔ Es una interfaz *Ethernet-Serie* integrada en un conector RJ45 (ver anexo A.2.3). El *Ethernet* es interpretado por el dispositivo y facilita la comunicación en *RS-232* (un estándar no oficial), ofreciendo varias señales en la *PCB*:
  - *Reset (In),* otorga la posibilidad de reiniciar la interfaz. Está conectada directamente el *microcontrolador*, por si en posteriores desarrollos fuese de utilidad.
  - *Data OUT,* representa la señal de transmisión de *Ethernet*, está conectada a *DI* de *U2*.
  - *Data IN,* conforma la señal de recepción de *Ethernet*, se encuentra en contacto con *RO* de U3.
  - *CP0, CP1, CP2,* ofrecen medios de control de flujo. Aunque no se utilizan, *CP0* y *CP2* están unidos por medio de una resistencia, que a priori constituirá un abierto, pero que en caso de presentar complicaciones puede cortocircuitarse para que el *RTS* (*CP0*) realice una llamada y el *CTS* (*CP2*) reciba una respuesta inmediata.

  Hay diversas alternativas frente a la utilización de una interfaz, como la implementación de la conversión Ethernet-Serie en la propia *PCB,* pero dado el bajo precio de la interfaz por la resolución de una labor un tanto complicada, no compensa desarrollar o buscar otro sistema conversor de estándares.
  Lantronix ofrece una opción asequible, muy depurada, robusta y simple de conectividad *Ethernet-Serie* (*Xport*). [14]

3.6.3 Bloque Conversor *RS485-RS232*

El bloque conversor *RS485-RS232* se encarga, obviamente, de la conversión *RS485-RS232*. Es un bloque integrado en todos los módulos de relés, acometiendo labores de adaptación de estándares de comunicación por medio de *transceivers*:

- *U5*, *U6* (*ADM1485ARZ*) ➔ Presentan una configuración similar a la anterior etapa. *U5*, controla el envío de datos del *microcontrolador* del módulo de relés (*MIC_RS485_TX*) a través de las líneas comunes de todos los módulos (*ETH_RS485_RX_A* y *ETH_RS485_RX_B*). Para evitar colisiones de datos *DE* está unido a una salida de control del *microcontrolador* (*MIC_RS485_TX_ENABLE*), de modo que solo habilita la transmisión diferencial mediante *A* (*RS485_TX_A)* y $\overline{B}$ (*RS485_TX_B)* cuando este se lo indica, y durante el resto del tiempo estas salidas (*A* y $\overline{B}$) se encuentran en un estado de alta impedancia. Aunque en reposo (*DE* en estado lógico bajo) el dispositivo opera como receptor, *RO* no está conexionado, por lo que cualquier salida permanece inalterable. [39]
  U6 transforma el protocolo *RS-485* diferencial (*RS485_RX_A* y *RS485_RX_B*, proveniente de *ETH_RS485_TX_A* y *ETH_RS485_TX_B* respectivamente) a *RS-232*, interpretable por el *microcontrolador* (*MIC_RS485_RX*).
  Aunque las huellas de los componentes permiten la utilización de numerosos *transceivers,* lo correcto es usar exactamente el mismo modelo que en la etapa anterior, ya que de otro modo se produciría un cuello de botella en las características más restrictivas de los modelos de emisor/receptor.

3.6.4 Bloque de *Microcontrolador*

Actualmente, de entre todas las tecnologías embebidas programables (consultar anexo A.3) predominan principalmente un grupo reducido constituido por dispositivos con características dispares:

- *Microcontroladores*, dedicados a usos de propósito general para procesamiento de la información y control de elementos externos. El esfuerzo de la implementación de sistemas integrados con esta tecnología se limita al desarrollo del software y su validación. El dominio de estos dispositivos está muy extendido, lo que facilita su integración.
  El rendimiento del sistema (especialmente los ciclos de reloj) está determinado por el código, es por ello que un código compacto con un uso eficiente de la arquitectura del *microcontrolador* es esencial.
- Procesadores digitales de señales (*DSPs*), que implementan las funciones básicas de muchos algoritmos de procesamiento de señal, lo que optimiza el uso de transistores y

ciclos de reloj para las operaciones necesarias. El código es más sencillo que en los *microcontroladores*.

- *FPGAs*, que localizan el trabajo de desarrollo en la codificación que requiere configurarlos. Necesitan de cierta especialización para ser implementados, aunque su uso está muy extendido. Sin embargo, presentan un alto nivel de redundancia de transistores, lo que aumenta su coste, una optimización de ciclos reloj limitada y mayor consumo de energía.
- *ASICs*, especialmente destinados a aplicaciones concretas, lo que conlleva un mayor aprovechamiento del número de transistores y los ciclos de reloj, que se traduce en menor consumo y un precio más bajo, pero a cambio requiere de mayor tiempo de desarrollo y una gran preparación de los recursos humanos.

Dada la naturaleza de la aplicación es conveniente el uso de dispositivos estandarizados de función programable secuenciales, *microcontroladores*, ya que las *ASICs* representan un coste asumible en grandes volúmenes de fabricación y las *FPGAs* suponen una alternativa adecuada para aplicaciones complejas (con numerosas interfaces, cuantiosas entradas y salidas de control, etc.) o altas velocidades.

Los *DSPs* son procesadores destinados fundamentalmente para la simplificación de algoritmos de procesamiento de señal con ciclos de reloj mínimos; sin embargo, los *microcontroladores* están diseñados para sistemas integrados con múltiples funcionalidades.

Los *microcontroladores* son excelentes candidatos para ámbitos poco intensos en transmisión, de cierta sencillez y saturados en control de procesos o elementos externos secuenciales. Poseen altas prestaciones a un bajo coste y su integración no implica extensas duraciones de la fase de desarrollo. Esto lo convierte en la opción más competente para implementar el sistema.

Aunque el marco de alternativas se ha reducido considerablemente, es imprescindible acotar la elección a un solo modelo dentro del campo de los *microcontroladores*, para lo que se realiza un análisis de los requisitos del sistema. Dadas las necesidades de ensayo y medida tras la implementación del sistema, conviene un modelo con memoria no volátil de múltiples grabaciones (*Flash*, *NVRAM*…), además un factor determinante son el número de salidas de control que debe ofrecer el dispositivo para gobernar todos los relés (hasta 52 relés en el módulo *IND5003*) unidas a las *I/Os* de comunicación, grabación, *LEDs*, circuito oscilador, etc. Es por ello que los criterios de búsqueda deben regirse a un *microcontrolador* de memoria Flash (muy empleada) y numerosas *I/Os* (100 pines).

Si se considera el uso de *latches,* decodificadores o similares como alternativa a *microcontroladores* con cuantiosos pines (ligeramente más caros) y por tanto abundantes *I/Os*, podría convertirse en una buena opción; sin embargo, representa una elección económicamente semejante (inclusión de varios circuitos integrados más) y desde el punto de vista de la reparación de la *PCB* en caso de avería (más compleja) y del espacio ocupado por los componentes (mayores dimensiones) supone un perjuicio.

Existen numerosas compañías que ofrecen modelos con dichas prestaciones, por lo que para delimitar aún más la búsqueda se apunta a *microcontroladores* para aplicaciones de baja

potencia asequibles. Entre todos ellos cabe destacar el *STM32F101V8T6* (ver anexo A.3.1.4.1), el modelo de 100 pines de menores prestaciones de *STM*. Aunque no es determinante, otorga 64kB de *Flash* y 10kB de *RAM*, más que suficiente para elaborar un código que satisfaga los requisitos, por lo que constituye una elección adecuada para este sistema. [35]

El *microcontrolador* (*U1)* es el circuito integrado programable ubicado en cada uno de los módulos de relés, encargado del control de los elementos externos, tales como el tratamiento de la información, los procesos de comunicación y el dominio de los *drivers*. Para llevar a cabo su cometido está configurado de la siguiente manera:

- Las 52 posibles salidas de control de los drivers están situadas de forma que presenten facilidad de conexionado en la *PCB*, eviten salidas problemáticas tras el reinicio (*PB3 – JTDO*, *PB4 – JNTRST*, *PA15 –JTDI, PB2 – BOOT1*) y permanezcan libres funcionalidades que puedan ser utilizadas en posteriores modificaciones (*PA11 – USART1_CTS*, *PA12 – USART1_RTS*). Las *I/Os* se organizan en puertos (*PORT A, PORT B, PORT C, PORT D* y *PORT E*) a los que hay que proporcionarles alimentación (*VDD* a *+3.3VDC* y *VSS* a *GND*).
- En el desarrollo de tarjetas con dispositivos *ARM* se suele utilizar un conector por defecto (*20-pin IDC*), que soporta *JTAG*, *Serial-Wire Debug* y *SWV*. En la hoja de características del *microcontrolador* se especifica en que patillas se incorpora la funcionalidad necesaria para estas tres tecnologías. Sin embargo, este conector ocupa demasiado espacio, por lo que es conveniente emplear otro conector más pequeño (*Molex 15-91-2100*) en la placa únicamente con las señales útiles (*PROG_BOOT, PROG_TX, PROG_RST, PROG_RX, PROG_ SWCLK*, *PROG_SWIO*, *+3.3VDC* y *GND*). De esta manera mantiene la compatibilidad con *JTAG*, *Serial-Wire Debug* y *SWV,* pero dado que el *Serial-Wire Debug* ofrece modos de depuración, grabación y prueba con tan solo dos pines (*PROG_ SWCLK* y *PROG_SWIO*), solo se implementará un adaptador con estas señales y la alimentación (*+3.3VDC* y *GND)*. [3] [37]
- *BOOT1* (*PB2*) y *BOOT0 (PROG_BOOT*) otorgan la posibilidad de operar en diversos modos, *User Flash memory, System memory* y *Embedded SRAM*, mediante la configuración de su estado lógico, *BOOT1* – X y *BOOT0* – 0, *BOOT1* – 0 y *BOOT0* – 1 y *BOOT1* – 1 y *BOOT0* – 1 respectivamente. Dado que el modo *Embedded SRAM* no es de interés para este sistema se cortocircuita *BOOT1* a 0, y puesto que el principal modo de uso es *User Flash memory* se emplea una resistencia de *pull down* y una resistencia limitadora de poco valor en serie para *BOOT0,* de forma que se eviten problemas debidos a ruidos o interferencias. (NRST) PROG_RST también podría provocar reinicios indeseados, es por ello que se utiliza un *pull up.* [37]
- El *microcontrolador* incorpora una manera de conservar el contenido de registros cuando la alimentación desaparece, el pin *VBAT* puede ser conectado a una tensión *standby* suministrada por una batería u otra fuente de energía. Este sistema no requiere esta funcionalidad, por lo que tal y como recomienda el fabricante es preciso conectar *VBAT* externamente a *VDD* (*+3.3VDC*), y como precaución ante grandes corrientes una resistencia limitadora en serie. [37]
- Las líneas *CFG_SW1, CFG_SW2, CFG_SW3* y *CFG_SW4*, conectadas a PC13, PC14, PC15 y PC1 respectivamente, establecen una forma de que el *microcontrolador* reconozca el

tipo de módulo de relés en el que se encuentra, de modo que un único código grabado en todos los *microcontroladores* se adapte a cada tipo de módulo.

- El circuito oscilador del *microcontrolador* marca el ritmo de trabajo de este. En el propia *datasheet* del modelo indica cómo y entre qué valores debe estar diseñado este circuito. El resonador, construido por dos condensadores conectados a masa y un cristal entre sus otros terminales, debe ir conectado entre las pines *OSC_IN* y *OSC_OUT*, de modo que un condensador esté unido a una patilla del cristal y *OSC_IN* y el otro condensador al otro terminal del cristal y *OSC_OUT*, permaneciendo las patillas restantes de los condensadores conectadas a masa (*GND*). El modelo empleado ofrece una frecuencia de trabajo de entre 4 y 16 MHz con la precisión adecuada (*HSE*), teniendo en cuenta que una frecuencia alta permite un mayor aprovechamiento de las capacidades del *microcontrolador*, pero no es adecuado acercarse al límite, una frecuencia de 12MHz es apropiada para el sistema. Acotado el valor del cristal (12MHz), las capacitancias de los condensadores son determinadas por la misma hoja de características del cristal empleado (*ABM8G-12000MHZ-18-D2Y-T*) y delimitadas por un rango (de 5pF a 25pF) y por recomendaciones en el *datasheet* del *microcontrolador.* [1] [36]
- La interfaz *Ethernet-Serie* dispone de un pin por el que puede ser reiniciada, si bien no goza de utilidad en el diseño, no está de más permitir resetear el dispositivo a través del *microcontrolador*. Para ello, se reserva una salida de control (*PC0*) que podría reiniciar la interfaz.
- Existen tres *LEDs*, verde, amarillo y rojo, que requieren de salidas de *microcontrolador* para controlar su encendido y apagado. Puesto que el *LED* verde señala una alimentación adecuada del sistema, solo serían necesarias dos salidas (*PC2* y *PC3*) con sus correspondientes resistencias, limitadoras de corriente, en serie.
- El bloque conversor *RS-485-RS232* transforma la comunicación del *microcontrolador* a RS-485 y viceversa. Acondiciona un canal de transmisión (*MIC_RS485_TX*), una línea de recepción (*MIC_RS485_RX*) y una habilitación de la transmisión (*MIC_RS485_TX_ENA*) para impedir colisiones de información de diferentes módulos. Es por ello que son imprescindibles tres *I/Os,* no obstante si son elegidos tres pines con dicha funcionalidad se ahorra posterior desarrollo en el código, por lo que *PB10* y *PB11* con funciones *USART3_TX* y *USART3_RX* respectivamente y por defecto, suponen una buena opción para realizar las comunicaciones. Solamente *MIC_RS485_TX_ENA* necesita una salida de control del *microcontrolador* cualquiera (*PE15*).
- Ante la numerosa cantidad de módulos de relés es indispensable establecer una manera de identificar cada módulo en concreto. Un direccionamiento, bien mediante un *switch* o mediante un posible *backplane*, es ideal para ello. Cinco líneas (*I/Os* del *microcontrolador*) son suficientes para registrar hasta 32 módulos de relés, el máximo soportado por el estándar *RS-485*.

3.6.5 Bloque de *Drivers*

Las salidas del *microcontrolador* destinadas a gobernar el comportamiento de los relés carecen de la capacidad de suministrar corrientes adecuadas para tal propósito. El bloque de drivers se ocupa de garantizar una corriente suficiente para la conmutación del relé cuando la salida del *microcontrolador* correspondiente lo indique.

Existe multitud de dispositivos aptos para este cometido, por lo que se aplican filtros adecuados para restringir las posibilidades:

- Encapsulado para *SMT* (*SO-16*).
- Abastecimiento de corriente sobrado y regido por el relé que necesite más corriente (*RTD14005F*), determinada por la hoja de características (la tensión a la que es sometida la bobina entre la menor resistencia que manifiesta, $62 \pm 10\%\ \Omega \simeq 56\Omega$), unos 90mA. [45]
- Compatibilidad con las salidas del *microcontrolador* a 3.3V.
- Alimentación a 5V.

Un circuito integrado y económico que cumple con estos requisitos es *ULN2003A,* un *array* de transistores *darlingtons* de alto voltaje y corriente, que contienen 7 pares *darlingtons* en colector abierto con emisores comunes.

Tal y como advierte el fabricante, cada par *darlington* es capaz de suministrar unos 350mA para un voltaje aplicado en la entrada de control de 3.3V. Además incluye protecciones (*Common free Wheeling diodes*) que deben conectarse a *Vcc*) para los picos de corriente provenientes de la carga (los relés). [41]

Como precaución ante ruidos e interferencias, es conveniente emplear *pull downs* en las entradas de control de los *ULN2003A*, de modo que se eviten conmutaciones indeseadas.

De esta manera, las bobinas de los relés, unidas en uno de sus extremos a 5V y otro a las salidas de estos circuitos integrados, son recorridas por suficiente corriente como para conmutar y cumplir su función.

Capítulo 4. Diseño e implementación del *software*

4.1 Procedimientos específicos del *software*

El desarrollo del *software* constituye una etapa que debe adaptarse a las especificaciones del diseño del *hardware*. Si bien el *software* debe implementar las medidas necesarias para que el usuario se comunique con el módulo de relés indicado de manera intuitiva, además de proporcionar una serie de comandos *VXI* (estándar *VISA*) que permitan una llamada de estos sencilla y previsible (similar a otros instrumentos *VXI*) para el control total de los relés, es necesario construir un *software* estructurado en varias capas:

- Capa del *firmware*, conformada por un lenguaje de "bajo" nivel (comparada con el resto de capas de *software*) que el *microcontrolador* es capaz de procesar.
- Capa del *driver*, constituida por comandos llamados desde la GUI, que transforman las acciones del usuario en la GUI a lenguaje interpretable por el *microcontrolador*, y mantienen cierta independencia con respecto a la GUI, de forma que el sistema puede ser controlado únicamente mediante estos comandos.
- Capa de la *GUI*, compuesta por un conjunto de objetos gráficos que suministran un entorno visual sencillo al usuario para el dominio del sistema y acotan la comunicación para aumentar la eficacia y evitar posibles errores. Sostiene una completa individualidad hacia la capa del firmware, ya que se comunica con ella mediante la capa del driver.

Esta configuración conforma una arquitectura en la que los elementos de una capa envían solicitudes a elementos de la capa inferior que deriva en una cascada de peticiones. Esto brinda una disposición por niveles con misiones simples con múltiples beneficios:

- Resulta más inteligible, cada capa es entendible sin considerar las otras.
- Se reduce la dependencia entre capas, lo que favorece su posible sustitución con implementaciones alternativas de los servicios fundamentales.
- Incrementa su escalabilidad, cada capa construida permite su inclusión en servicios de mayor nivel.
- Facilita la estandarización de servicios.

Una producción eficiente y eficaz del producto *software* requiere del establececimiento un modelo para el desarrollo de *software*. En este caso, dado que se trata de un sistema en el que intervienen varias capas de *software,* es conveniente la validación de las versiones de *software,* en el mejor de los casos una vez conseguido el prototipo *hardware,* mediante una depuración constante de porciones de funcionalidad de cada capa, es decir, se ensayan fracciones de código, que constituyen piezas funcionalmente independientes, sobre el *hardware* implementado.

Ahondando en la evolución del *software*, también es preferible seguir un orden para acrecentar la sencillez en la consecución de objetivos y la finalización con éxito del proyecto. El

diseño de la interfaz gráfica debe acomodarse al planteamiento del *driver*, y este a su vez debe adecuarse al diseño del *firmware,* pero no elimina la posibilidad de que se ejecuten posteriores modificaciones en cualquiera de las capas. Lo que implica la implementación en orden consecutivo del proyecto (*Firmware* ➔ *Driver* ➔ Interfaz gráfica).

4.2 *Firmware*

El *firmware* es una agrupación de instrucciones en código máquina con cometidos particulares, grabado en la memoria no volátil del *microcontrolador*, la combinación de la memoria y el código de programa y los datos almacenados en ella, y establece la lógica de bajo nivel que gobierna los circuitos electrónicos del *microcontrolador* y a su vez del sistema.

Para lograr lograr una buena elaboración de este código es necesario encontrar un entorno de desarrollo adecuado y realizar una simulación del código implementado, acorde con el *microcontrolador* utilizado (concretamente el *STM32F101V8,* un *ARM Cortex-M3 MCU* con 64 Kbytes de *Flash* y 36 MHZ de *CPU*). Esto restringe la búsqueda a unos pocos programas de desarrollo, entre los que destaca *MDK-ARM* de *Keil.*

4.2.1 *Software MDK-ARM*

El *Microcontroller Development Kit* (*MDK*) es un completo entorno de desarrollo para los dispositivos basados en los procesadores *Cortex-M, ARM7, ARM9* y *Cortex-R4*, aunque está especialmente diseñado para aplicaciones con *microcontroladores*, combinando el compilador *C/C++* de *ARM* con el sistema operativo a tiempo real *RTX* de *Keil* y librerías intermediarias entre capas.

Las herramientas se encuentran integradas en *µVision* que ofrece un gestor de proyectos, un editor, un depurador y un simulador en un entorno de desarrollo fácil de usar, es por ello que es el entorno de desarrollo integrado en el que se implementa el *firmware*. El ciclo de desarrollo del *software* comienza con la creación de un proyecto y su configuración.

En las opciones del proyecto (*Options for Target*) se incluyen diversos apartados en los que es necesario establecer algunos parámetros para poder disfrutar de más aspectos del entorno de desarrollo (depuración y simulación):

- Pestaña *Device*, permite la selección del modelo del *microcontrolador* empleado (*STM32F101V8*).
- Pestaña *Target*, facilita la introducción de la frecuencia del cristal empleado (*12MHz*).
- Pestaña *Output*, posibilita seleccionar la creación de ficheros de grabación (*HEX*), de información de depuración y de información de exploración.

- Pestaña *C/C++,* otorga la capacidad de interactuar con el preprocesador, con el fin de realizar sustituciones , procesar directivas y analizar el fichero fuente antes de la fase de compilación real. En el caso del sistema conviene indicar en el campo *Define*:
    - *USE_STDPERIPH_DRIVER*, que admite el uso de las funciones controladoras de periféricos de *ST*, lo que ahorra tiempo en el desarrollo.
    - *STM32F10X_MD_VL*, que indica al fichero *stm32d10x.h* el uso de un dispositivo de densidad media.
    - *HSE_VALUE = 12000000,* que señala la programación del *RCC* para el *HSE* a *12MHz*, lo que evita modificar todas las funciones que emplean el valor en caso de cambiarlo.
- Pestaña *Debug*, ofrece la elección del depurador usado en el sistema, *ST-Link Debugger*, que brinda una interfaz *SWD/JTAG* para comunicarse con el *microcontrolador* de la familia *STM32*. [38]
- Pestaña *Utilities*, permite escoger el método de programación de la memoria flash, concretado en este supuesto al *ST-Link Debugger*.

El uso del paquete *STM32F10x_StdPeriph_Driver* convierte la tarea de programación en una tarea más sencilla, pues además de albergar ejemplos e información de programación con *microcontroladores* de la línea *STM32F10x*, incorpora ficheros insertables al proyecto que resultan de utilidad en la programación de los periféricos: *stm32f10x_conf.h* y *stm32f10x_conf.c* (ambos indispensables para la configuración de los controladores de periféricos como *USART, GPIOs RCC,* etc.), *startup_stm32f10x_md.s* (necesario para la simulación de *microcontroladores* de densidad media) y *stm32f10x.h* (contiene todas las definiciones de los registros de los periféricos, definiciones de bits y mapeo de memoria).

Conformados los parámetros de configuración, conviene continuar con la implementación de los ficheros fuente y cabeceras que entreguen la funcionalidad al sistema, que se encuentran adoptando una estructura en la que existe un programa principal que realiza llamadas a rutinas para desempeñar el cometido de cada módulo de relés.

4.2.2 Programa principal

El programa principal (*main*) actúa como punto de partida, controla la ejecución del programa y realiza llamadas a otras funciones del *firmware*.

Tras un reinicio del *microcontrolador*, este garantiza la correcta configuración del módulo de relés en el que se encuentra incorporado, para lo que inicializa la tarjeta en partes diferenciables:

1. *Inicializar IOs*. Conforma cada una de las *IOs* con sus correspondientes características.
2. *Inicializar relés y configuración del módulo*. Establece una clasificación del módulo en cuanto a dirección (proporcionada por el *switch* o un posible *backplane*) y tipo de tarjeta (suministrado por una serie de conexiones en cada módulo a la alimentación o

tierra), asignando a cada bit de estas entradas de catalogación un peso diferente; también resetea todos los relés y por consiguiente sus estados.

3. *Inicializar comunicación Serie*. Configura la comunicación serie de acuerdo a unos parámetros predeterminados entre la interfaz Ethernet-Serie y el módulo de relés.

Tras fijar las propiedades necesarias para el funcionamiento del sistema, se procede al control de la aplicación. Dada la naturaleza del servicio facilitado, el programa debe implementar una comprobación de la comunicación continua, para lo que se emplea un bucle infinito en el que cada carácter recibido se analiza y se procesa. En este bucle también se comprueba cuanto tiempo llevan encendidos el *LED* rojo y el *LED* amarillo, y en caso de que alguno supere las 1000 unidades (valor deducido experimentalmente) debe apagarse. Aunque estas unidades de tiempo no suponen un valor constante ya que dependen de la llamada y consecución de funciones, la construcción de un algoritmo más complejo (uso de interrupciones) constituye una pérdida de tiempo, pues la implementación actual presenta una total consecución de objetivos en cuanto a funcionalidad (los *LEDs* amarillo y rojo señalan claramente y de manera visible, la existencia de comunicación y fallos, respectivamente).

La figura 16 ilustra el diagrama de actividades del programa principal de modo claro y conciso.

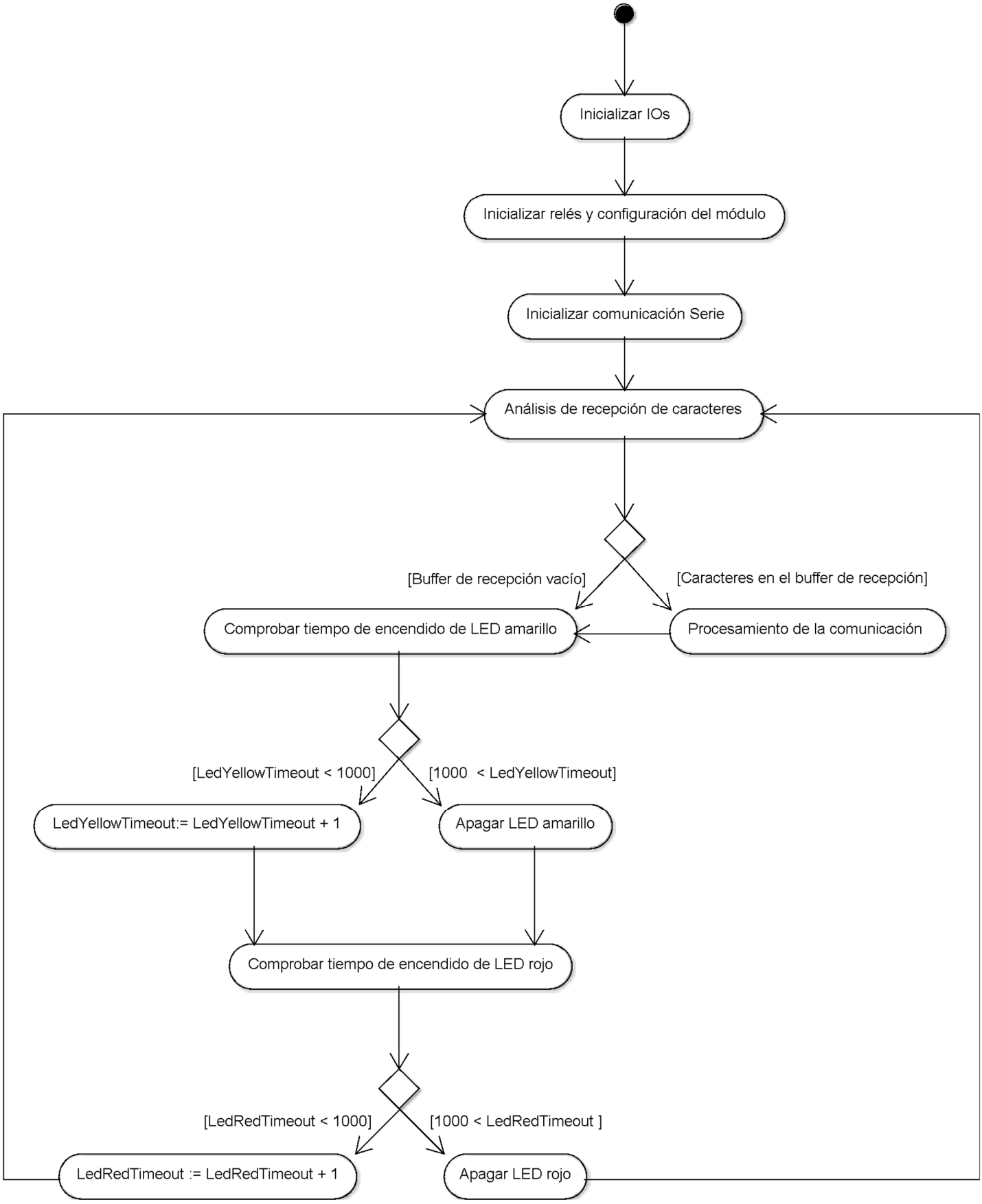


Figura 16. Diagrama de flujo del programa principal

4.2.3 Configuración de *IOs*

El *microcontrolador* se sirve de periféricos de entrada/salida (*IOs*) para relacionarse con el exterior, tanto para obtener información a través de ellos, como para interactuar con ellos (ver anexo A.3.1.1.4.2.3). La configuración de las entradas/salidas del *microcontrolador* constituye el primer paso hacia la consecución de un *firmware* funcional; cada una de las *IOs* debe conformarse de acuerdo a la distribución establecida en el diseño del *hardware*.

Si bien, conformar las propiedades de las entradas y salidas del sistema se presenta como una labor complicada, *STM32F10x_StdPeriph_Driver* adjunta código de ejemplo que facilita la inicialización de las entradas y salidas del *microcontrolador,* así como las librerías necesarias para emplear las funciones de *ST* en la configuración de periféricos. Se puede observar que aunque es posible tratar cada pin del *microcontrolador* individualmente, es más eficiente fijar las características de grupos de entradas o salidas que las compartan, por lo que es conveniente establecer una serie de pasos para la configuración de *IOs*, tal y como muestra la figura 17. [37]

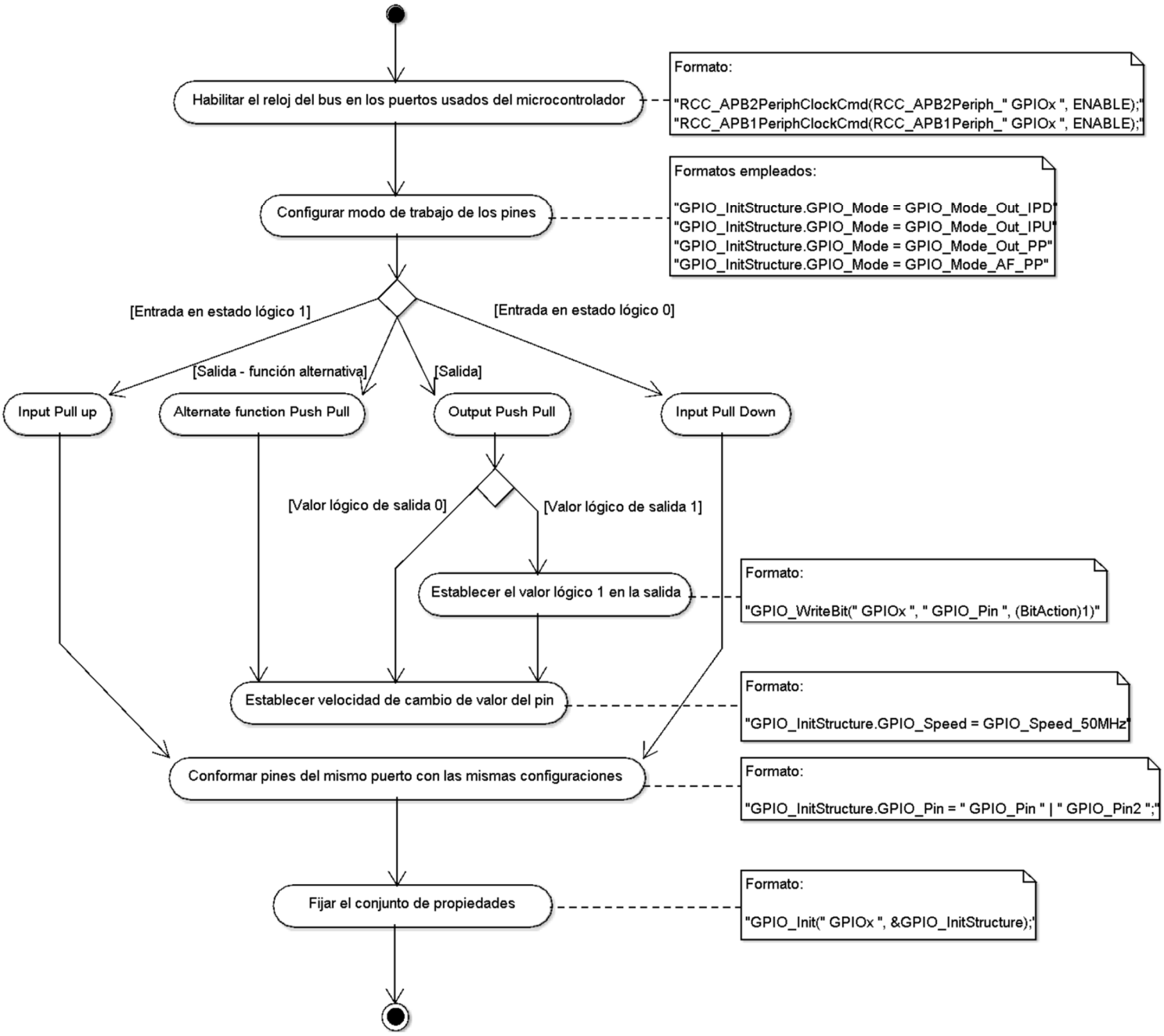


Figura 17. Diagrama de flujo de la Configuración de *IOs*

Como se aprecia en el diagrama anterior, primero se habilita el reloj del bus en los puertos de los pines del *microcontrolador*, y dado que todas las entradas y salidas empleadas en el sistema (excepto los pines con funcionalidad *USART3*) se encuentran conectadas a un bus de alta velocidad (*APB2*), se utiliza el reloj en este.

Tras esta instrucción, es indispensable elegir el modo de funcionamiento de los pines. EL microcontrolador ofrece diversos modos (*GPIO_Mode_AIN, GPIO_Mode_IN_FLOATING, GPIO_Mode_IPD, GPIO_Mode_IPU, GPIO_Mode_Out_OD, GPIO_Mode_Out_PP, GPIO_Mode_AF_OD* y *GPIO_Mode_AF_PP*) [37]; sin embargo, dado que no se emplea ninguna entrada analógica (*GPIO_Mode_AIN*), las salidas *Open Drain* (*GPIO_Mode_Out_OD* y *GPIO_Mode_AF_OD*) no ofrecen suficiente corriente para los relés, y no es recomendable el uso de entradas flotantes (*GPIO_Mode_IN_FLOATING*), las opciones se reducen a cuatro posibilidades:

- *GPIO_Mode_Out_PP*, es un modo de funcionamiento de salida con corrientes entrantes y salientes, idónea para el control de los *drivers* de los relés, de los *LEDs*, del reinicio de la interfaz *Ethernet-Serie* y el pin de habilitación de la transmisión: *GPIO_Pin_3, GPIO_Pin_4, GPIO_Pin_5, GPIO_Pin_6* y *GPIO_Pin_7* del puerto A, *GPIO_Pin_0, GPIO_Pin_1, GPIO_Pin_5, GPIO_Pin_6, GPIO_Pin_7, GPIO_Pin_8, GPIO_Pin_9, GPIO_Pin_12, GPIO_Pin_13, GPIO_Pin_14* y *GPIO_Pin_15* del puerto *B*, *GPIO_Pin_0, GPIO_Pin_2, GPIO_Pin_3, GPIO_Pin_4, GPIO_Pin_5, GPIO_Pin_10, GPIO_Pin_11* y *GPIO_Pin_12* del puerto *C*, *GPIO_Pin_0, GPIO_Pin_1, GPIO_Pin_2, GPIO_Pin_3, GPIO_Pin_4, GPIO_Pin_5, GPIO_Pin_6, GPIO_Pin_7, GPIO_Pin_8, GPIO_Pin_9, GPIO_Pin_10, GPIO_Pin_11, GPIO_Pin_12, GPIO_Pin_13, GPIO_Pin_14* y *GPIO_Pin_15* del puerto *D, GPIO_Pin_0, GPIO_Pin_1, GPIO_Pin_2, GPIO_Pin_3, GPIO_Pin_4, GPIO_Pin_5, GPIO_Pin_6, GPIO_Pin_7, GPIO_Pin_8, GPIO_Pin_9, GPIO_Pin_10, GPIO_Pin_11, GPIO_Pin_12, GPIO_Pin_13, GPIO_Pin_14* y *GPIO_Pin_15* del puerto *E*.
- *GPIO_Mode_IPD*, conforma una conducta de entrada con *pull down*, ideal para entradas en estado lógico de reposo 0. Se usa en las entradas que suministran la dirección del módulo de relés (provenientes del *switch*): *GPIO_Pin_6, GPIO_Pin_7, GPIO_Pin_8* y *GPIO_Pin_9,* del puerto *C,* y *GPIO_Pin_8* del puerto *A*.
- *GPIO_Mode_IPU*, ofrece la modalidad de entrada con *pull up*, conveniente para entradas en estado lógico de reposo 1. Se utiliza en el resto de entradas, de clasificación del tipo de módulo, de recepción de la comunicación (*RX*) y en *IOs* no aprovechadas: *GPIO_Pin_0, GPIO_Pin_1, GPIO_Pin_2, GPIO_Pin_11, GPIO_Pin_12* y *GPIO_Pin_15* del puerto *A*, *GPIO_Pin_2, GPIO_Pin_3* y *GPIO_Pin_4* del puerto *B*, *GPIO_Pin_1, GPIO_Pin_13, GPIO_Pin_14* y *GPIO_Pin_15* del puerto *C*.
- *GPIO_Mode_AF_PP*, proporciona la salida con función alternativa establecida por el fabricante. Es empleada únicamente en el pin de transmisión (*TX), GPIO_Pin_10* del puerto *B.*

En caso de tratarse de una salida en la que se necesita un estado lógico alto desde un primer momento, como el pin de control de reinicio (*Reset* con estado lógico 0) de la interfaz

Ethernet-Serie (*EXTERNAL_RESET_IN, GPIO_Pin_0* del puerto *C*), es inevitable fijar la salida antes de su configuración al estado lógico 1.

El *microcontrolador* ofrece varias velocidades de trabajo seleccionables de las salidas (10MHz, 2MHz y 50 MHz), pero dado que se desconoce la velocidad de cambio de las salidas de control, excepto en el supuesto de la rapidez del pin de transmisión (*TX*), es conveniente emplear la máxima velocidad posible (50MHz). [37]

Determinadas tanto las propiedades de los pines, como estos, es necesario organizarlos en puertos, de manera que solo es posible configurar los pines, de acuerdo a las especificaciones, de un solo puerto a la vez; es por ello que resulta beneficioso agrupar los pines en cuanto a puertos y atributos, para en un último paso fijar los rasgos de todos los pines de un mismo puerto con una sola instrucción.

4.2.4 Comunicación Serie

Universal Synchronous Asynchronous Receiver Transmitter (*USART*) es un puerto serie síncrono asíncrono emisor y receptor, con múltiples características para un amplio rango de protocolos serie (consultar anexo A.3.1.2.9).

El modelo de *microcontrolador* (*STM32F101V8*) empleado dispone de 3 *USARTs*. *USART1* se encuentra comunicada por un bus de alta velocidad (*APB2*, con una velocidad de hasta 72MHz), mientras que *USART2* y *USART3* utilizan un bus de baja velocidad (*APB1*, operando a 36MHz), pero la aplicación de este sistema no requiere de altas frecuencias de comunicación. Así, dado que cualquier USART es válido, se ha elegido el *USART3* en la fase de desarrollo de hardware, por su ubicación respecto a su relación con otros elementos. [37]

El puerto *USART* es un periférico complejo con una gran variedad de opciones de configuración y registros, como se aprecia en la siguiente tabla:

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USART_SR**	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	**USART_DR**	Reserved																						DR[8:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	**USART_BRR**	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USART_CR1**	Reserved																		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USART_CR2**	Reserved																	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																		0	0	0	0	0	0	0		0	0		0	0	0	0
0x14	**USART_CR3**	Reserved																					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x18	**USART_GTPR**	Reserved																GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 2. Mapa de registros *USART* [37]

La comunicación serie constituye un módulo de software significativo dentro del sistema. Las rutinas *USART* pueden conformarse únicamente sobre los periféricos *USART*; sin embargo, presentan un desaprovechamiento de los recursos por sí solas, por lo que es conveniente el uso de interrupciones. Las interrupciones pueden enlazarse con los bits de los registros de la tabla 2, de manera que cualquier modificación en su valor permita ser advertida al momento, mediante la interrupción, tal y como muestra la siguiente tabla:

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
CTS flag	CTS	CTSIE
Transmission complete	TC	TCIE
Received data ready to be read	RXNE	RXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
Break flag	LBD	LBDIE
Noise flag, Overrun error and Framing error in multibuffer communication	NE or ORE or FE	EIE(1)

Tabla 3. Solicitudes de Interrupción *USART* [37]

Como se puede observar en la tabla 3, existen numerosas fuentes de disparo de interrupciones, cada una de ellas para distintos propósitos. Para el uso de interrupciones en estos bits, es necesaria la habilitación en el registro de control *USART* (*USART_CR1*). También

es imprescindible habilitar un canal *NVIC USART_IRQn* para aplicar la interrupción a su rutina de servicio. El *Nested Vectored Interrupt Controller* (*NVIC*) es una parte esencial del procesador *Cortex*, facilita un vector para todos los disparos de interrupciones y la rutina de servicio debe descubrir qué interrupción ha disparado un evento, para lo que se observan los *flags* (bits) *USART* del registro de estado (*USART_SR*). [37]

Las interrupciones realmente útiles para la comunicación del sistema, el disparo de la transmisión y/o la recepción, se reducen únicamente a tres [37]:

- *USART_IT_RXNE*, dedicada a la recepción, alcanza el estado lógico alto cuando la información del registro de desplazamiento de recepción de datos se traslada al registro de datos *USART_DR*, es decir, cuando el registro de datos no está vacío.
- *USART_IT_TXE*, destinada a la transmisión, modifica su valor lógico a 1 cada vez que la información del registro de datos es transferida al registro de desplazamiento, lo que implica la disposición de entrega de otro byte, o sea, un registro de datos de transmisión vacío.
- *USART_IT_TC*, empleada para indicar el momento en el que se ha completado la transmisión.

La configuración del canal NVIC para este caso resulta una labor sencilla:

- Canal *USART3_IRQn*, dado que es la *USART* que se emplea.
- Prioridad y subprioridad 0, pues no existen interrupciones que puedan concurrir en el mismo instante y comprometer la funcionalidad del sistema.

La comunicación basada en interrupciones resulta ligeramente impredecible y peligrosa, dado que es posible que ocurran interrupciones durante procesos de envío o recepción y remitan a estos mismos procesos. Para subsanar estos paradigmas es indispensable la creación de un doble *buffer*, de forma que las rutinas de interrupción almacenen o transmitan la información en él. El doble *buffer* consiste en dos *buffers*, cada uno con un cometido, uno se encarga de los datos a ser transmitidos y el otro se ocupa de la información recibida.

Para este sistema, un buen *buffer* se encuentra implementado con un algoritmo de cola (estructura tipo *FIFO*), la salida de los datos ocurre en el orden en que llegan. El *buffer* se comporta como una cola circular, y para ello contiene dos punteros o valores:

- Puntero o valor de lectura (*rdIdx*), que indica el lugar de lectura actual y aumenta de uno en uno cuando se produce la lectura del dato.
- Puntero o valor de escritura (*wrIdx*), que señala el punto de escritura presente e incrementa en una unidad cada vez que se incorpora un dato en el *buffer*.

De modo que la diferencia absoluta entre estos valores ($count = |wrIdx - rdIdx|$) identifica un *buffer* vacío en caso de ser nula, y un *buffer* lleno para el caso de igualar al tamaño del *buffer* (el valor de lectura se encuentre al menos una posición por delante del valor de escritura).

Aunque establece una solución frente a los problemas con las interrupciones, cabe la posibilidad de sufrir un *overflow*, un derrame del *buffer*, para lo que es inexcusable asegurar

un buen tamaño (256 bytes) y un método de eludir que el valor de escritura alcance el de lectura.

El puerto *USART* del *microcontrolador* establece pines para la comunicación serie con funciones alternativas por defecto, aunque solo son indispensables el envío (*USART3_TX*) y recepción (*USART3_RX*), ya que no se emplea una comunicación síncrona (*USART3_CK*) ni controladores *hardware* de flujo (*USART3_CTS y USART3_RTS)*. Los pines de envío y recepción necesitan de la configuración de sus parámetros, pero resulta una tarea muy simple:

- Habilitación del reloj del bus *APB1*.
- Pin Receptor *RX*:
    - Modo de operación como *Input Pull Up*, ya que su estado es controlado por otro dispositivo (externo).
    - No es necesario establecer la velocidad, ya que se trata de una entrada.
- Pin Transmisor *TX*:
    - Modo de trabajo como *Alternate Function Push Pull*, pues el pin de transmisión (salida) presenta la funcionalidad como tal en una función alternativa con corrientes tanto entrantes como salientes.
    - Velocidad de operación de 2MHz. La velocidad de transmisión estará determinada por los baudios establecidos, muy lejos de los 2MHz, es por ello que es una buena elección.

La comunicación entre dispositivos debe presentar las mismas características, ya que de otro modo, sería imposible el entendimiento entre emisor y receptor. Estos rasgos han de conformarse de acuerdo a las aplicaciones del sistema:

- Velocidad de comunicación (*baud rate*) de 115200 baudios, suponen un intercambio de información rápido y relativamente cercano al límite establecido por la interfaz Ethernet-Serie (921600 baudios), más que suficiente para el propósito del sistema.
- Longitud de palabra de 8 bits, indica el número de bits transmitidos en un paquete de datos, es la longitud más empleada, corresponde al estándar *RS-232.*
- 1 Bit de parada, indica el final de la transmisión, es un valor muy empleado, también por el estándar *RS-232*.
- Sin paridad, no es necesaria, por lo que no se emplea.
- Sin control *hardware* de flujo, pues resulta redundante e ineficaz en el caso propuesto.

Dado que la comunicación del *microcontrolador* necesita inicializar la configuración serie, es irremediable establecer un procedimiento que permita restaurar los valores en cada reinicio del *microcontrolador*. Se trata de establecer las propiedades apropiadas para cada elemento que interviene en la comunicación serie (el canal *NVIC*, los pines empleados para la recepción y transmisión, los valores determinantes de la comunicación, interrupciones, etc.); si bien, tal y como muestra la figura 18, la mayoría de actuaciones pertinentes se encuentran completamente detalladas, se puede apreciar que la configuración de los pines guarda grandes semejanzas con el modelado de funcionamiento del resto de *IOs*, por lo que es posible emplear el algoritmo ya empleado anteriormente (figura 17).

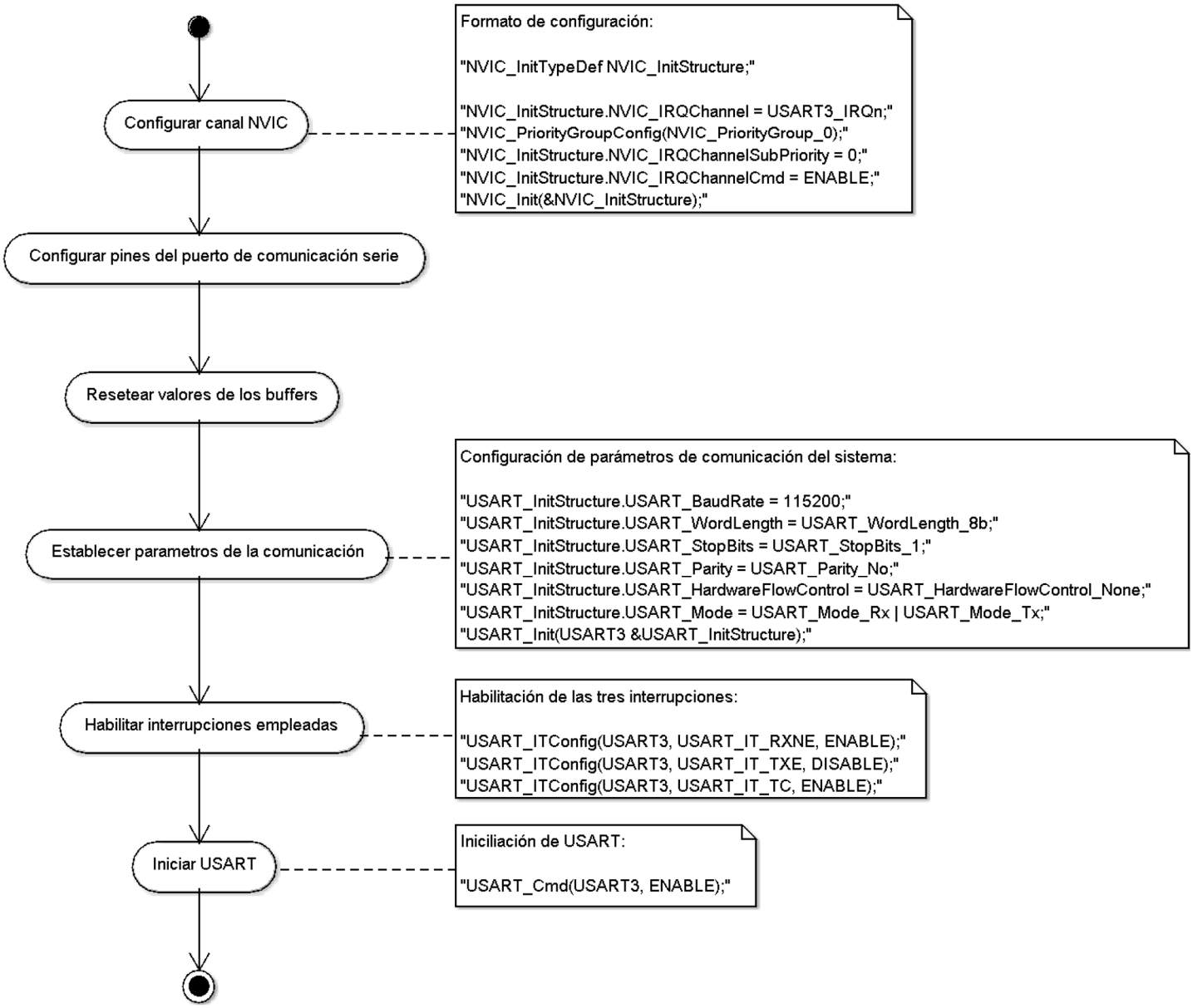


Figura 18. Diagrama de flujo de Inicialización de la comunicación serie

La habilitación y configuración de todos estos parámetros constituye una implementación de código que, además de permitir una comunicación correcta entre dispositivos, posibilita la programación de una subrutina de manejo de interrupciones (*IRQHandler*).

En la figura 19 se muestran las intervenciones que facilitan la inclusión y extracción de caracteres en el doble *buffer*, para su posterior tratamiento, desde la función *USART3_ IRQHandler*.

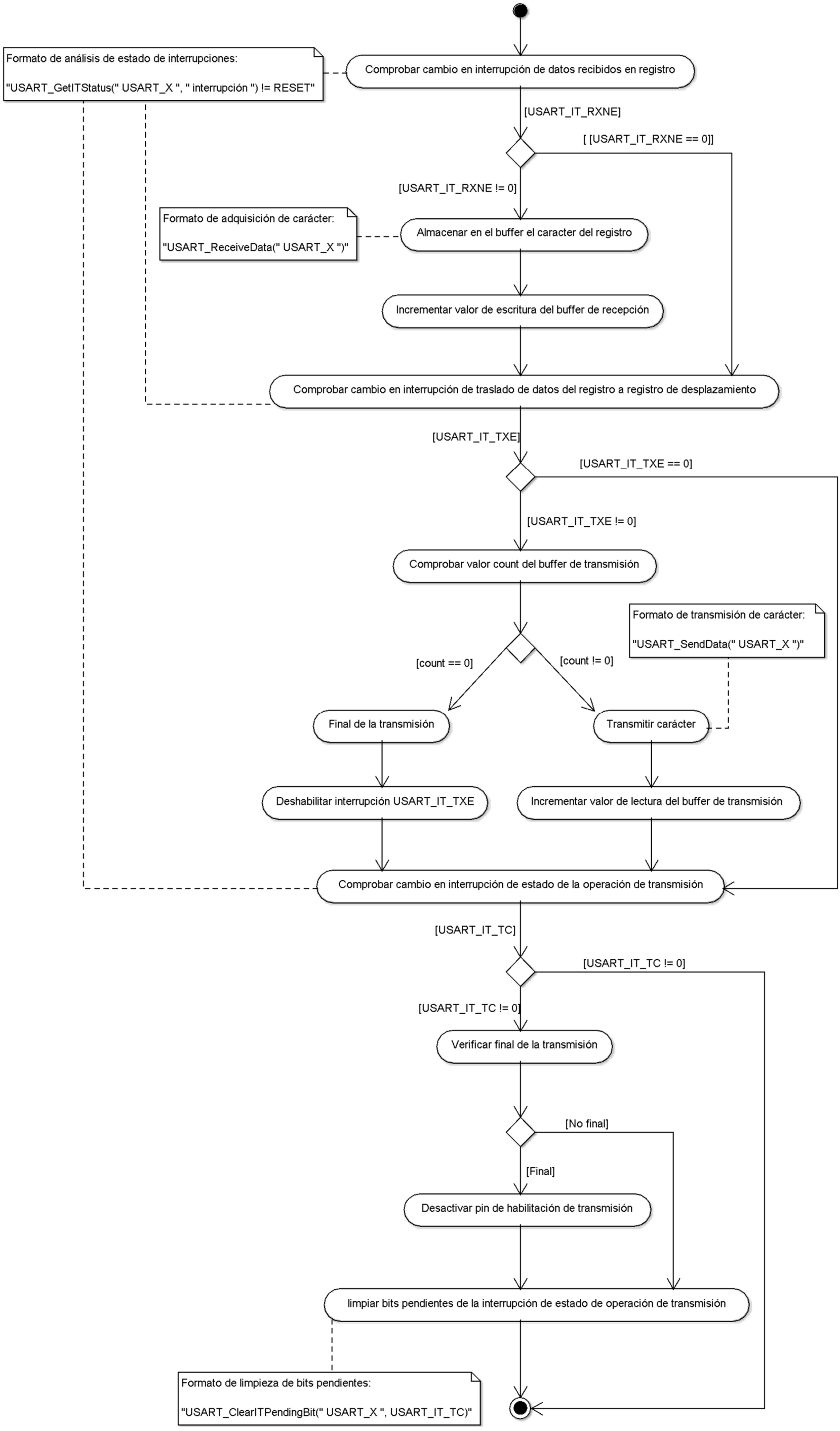


Figura 19. Diagrama de flujo del manejo de las interrupciones (*USART3_IRQHandler*)

En la función *USART3_IRQHandler* se emplean las interrupciones citadas. En primer lugar, se comprueba si el registro de datos de recepción está vacío (*MIC_RS485_RX* no ha aportado ningún dato) mediante la interrupción *USART_IT_RXNE*, y de no ser así se transfiere el carácter al *buffer* de recepción, aumentando el índice de escritura de dicho *buffer*. Después se analiza si el registro de datos de transmisión se encuentra libre de datos, y en ese caso se examina la diferencia entre los valores de escritura y lectura (*count*) del buffer de transmisión. De modo que en el supuesto de un buffer de transmisión con caracteres por enviar (*count* ≠ 0), se retransmiten por el pin *USART* correspondiente (señal *MIC_RS485_TX*) y se incrementa el índice de lectura de este buffer por cada uno de ellos, y en el caso contrario (*count* = 0) se da por finalizada la transmisión, por lo que se deshabilitala interrupción *USART_IT_TXE*, ya que de otra manera volvería a repetir el bucle. Por último, a través de la interrupción *USART_IT_TC* se estudia si el proceso de transmisión ha concluido, y si no resta más información por enviar se detiene la habilitación de la transmisión (*MIC_RS485_TX_ENA*); para terminar se limpia el bit de dicha interrupción.

La relación de lectura y escritura de caracteres con el resto del código constituye una tarea imprescindible para facilitar las operaciones de envío y recepción de información. La figura 20 presenta el algoritmo de recepción de caracteres del *buffer*, que consiste en el análisis del *buffer* de recepción.

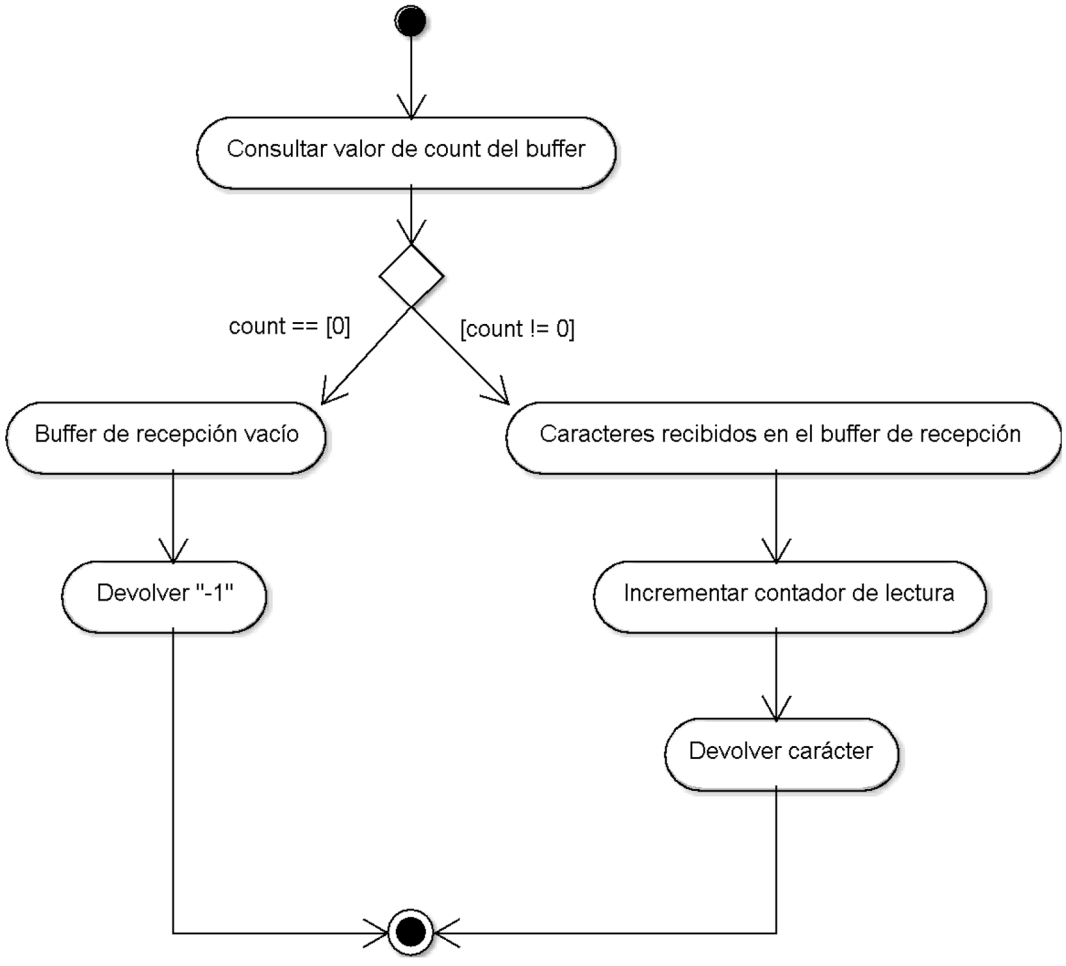


Figura 20. Diagrama de flujo de la recepción de caracteres

Si existe diferencia entre los índices de escritura y lectura (*count*) entonces significa que el *buffer* de recepción alberga algún carácter, es por ello que es leído ,y por consiguiente crece el valor de lectura, y devuelto como valor de retorno por la función. Si no fuese así (*count* = 0), la función retorna un "-1", de forma que ambos casos estén representados por un valor específico.

La figura 21 señala las actuaciones necesarias de la subrutina de envío de caracteres:

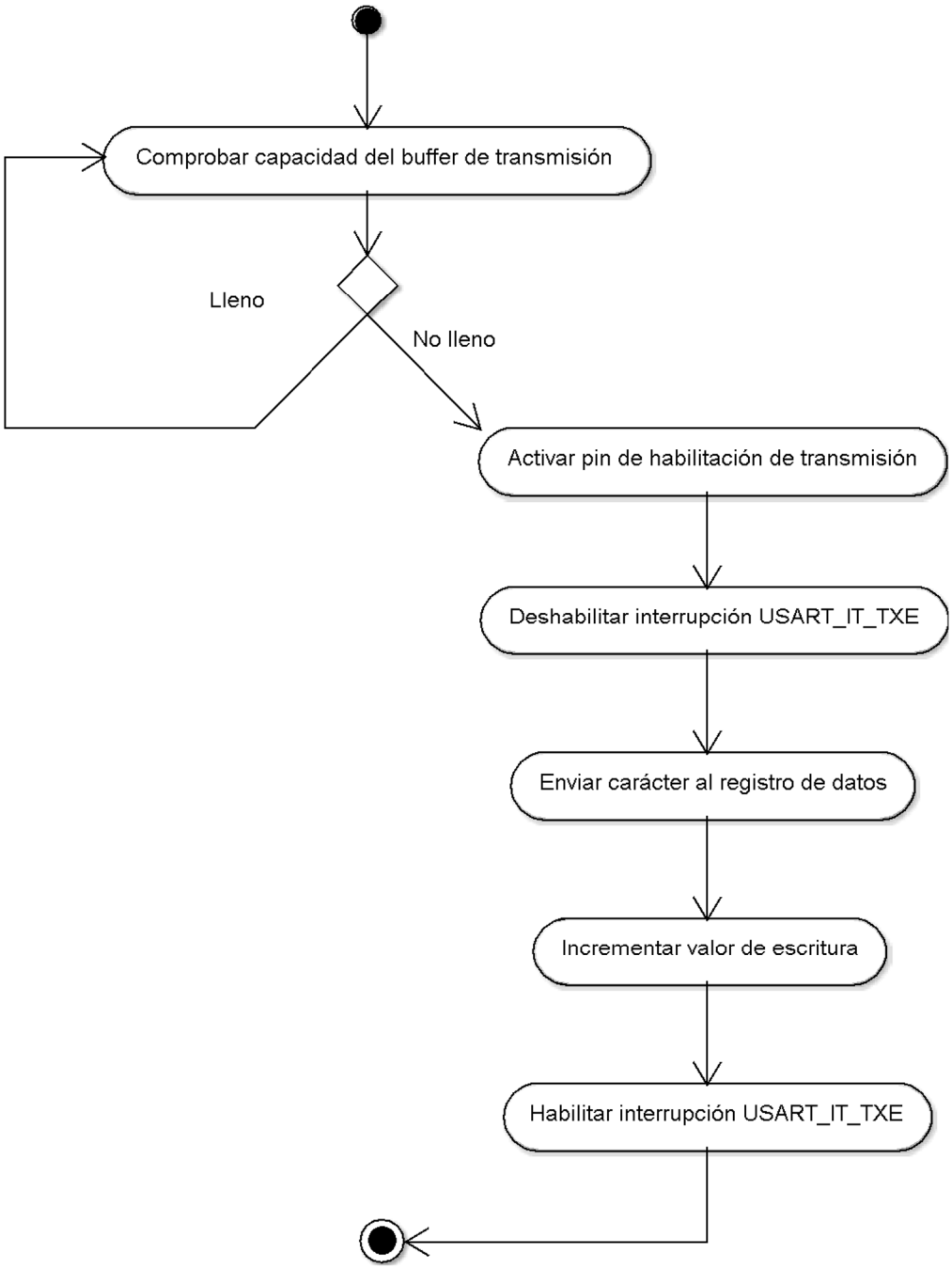


Figura 21. Diagrama de flujo del envío de caracteres

Se comienza con un estudio del estado del *buffer* de transmisión, de manera que si la posición del índice de lectura se encuentra tan solo a una unidad del valor de escritura, se sobreentiende que el *buffer* de transmisión está lleno, no es conveniente incorporar otro carácter al buffer, pues ocuparía la posición de otro carácter que aún no ha sido procesado. Para ello, se implementa un bucle que no termina hasta que el buffer no esté lleno, es entonces cuando se habilita la transmisión (señal *MIC_RS485_TX_ENA*), se deshabilita la interrupción *USART_IT_TXE* para evitar un envío prematuro, y se traslada el carácter al registro de datos, con el consecuente aumento del índice de escritura del buffer de transmisión. Para finalizar, se habilita la interrupción *USART_IT_TXE*, que es detectada inmediatamente por la interrupción y se transmiten los datos.

Así, el envío y admisión de caracteres, resulta una labor menos compleja; sin embargo, el uso de funciones de alto nivel, concretamente toda la familia de *printf()* (*fwrite(), fput(),* etc.) en la transmisión, simplifica aún más el código y su entendimiento, por lo que es conveniente emplearlo. Para llevar a cabo esta tarea es necesario redefinir las funciones de bajo nivel de las librerías, de modo que hay que determinar una versión propia de *__FILE* e implementar cada función empleada (*fputc(), fgetc(), ferror()* y *_ttywrch()*) de acuerdo al código (llamadas a las subrutinas de envío de caracteres y recepción de caracteres).

Aunque no representa utilidad de cara a la funcionalidad, es posible configurar el resto de puertos *USART* de la misma manera que se ha configurado el *USART3*, por lo que de cara a una futura redistribución de componentes y/o pines, no sobra su desarrollo.

4.2.5 Procesamiento de la información recibida

Cada módulo de relés es capaz de transmitir mensajes a través de la interfaz *Ethernet-Serie*, pero se ha implementado la comunicación de manera que solo existe una respuesta por parte del módulo en caso de que haya recibido instrucciones con anterioridad.

El programa principal comprueba en cada iteración la existencia de caracteres en el buffer de recepción, de modo que en el supuesto de que así sea, realiza una llamada a una función que se encarga de procesar cada carácter recibido.

El procesamiento de la información requiere de un conjunto de comandos específicos para recolectar y transmitir datos, de modo que al enviar estos comandos, los módulos de relés reproduzcan una réplica adecuada. Dado que el sistema debe presentar cierta compatibilidad con *VXI*, es conveniente emplear formatos similares a los comandos generales comunes descritos por el estándar *IEEE-488.2*, realmente útiles en el sistema (**IDN?, *RST* y **TST?*). Sin embargo, dado que el conjunto de módulos constituye un único recurso *VXI* al que referirse (una interfaz *Ethernet-Serie* para 32 módulos) es indispensable conformar comandos generales que incluyan la dirección del módulo con el que se desea establecer la comunicación, es por ello que se adaptan los comandos marcados por el estándar *IEEE-488.2* a **IDNxx?*, **RSTxx* y **TSTxx?*, donde xx representa la dirección del módulo con dos dígitos.

Si bien los comandos se encuentran correctamente determinados, presentan desventajas en el tratamiento de datos, ya que incorporan la dirección del módulo al final del comando, cuando cada módulo debería conocer cuanto antes si debería responder o no a la llamada. Para ello, los módulos de relés también responden a los formatos *Mxx*IDN?, Mxx*RST Y Mxx*TST?*, en los que xx también corresponde a la dirección del módulo expresada con dos dígitos. Dado que este procesamiento de caracteres resulta ligeramente más beneficioso, las acciones pertinentes a los relés, la conmutación de un relé a una posición y la consulta del estado de un relé, conviene expresarlas en los formatos *MxxRyyPz* y *MxxRyyP?* respectivamente, donde *xx* simboliza la dirección del módulo con dos dígitos, *yy* el número de relé con dos dígitos y *z* la posición del relé con un solo dígito.

La figura 22 ilustra el diagrama de flujo del procesamiento de la información de acuerdo a los comandos instaurados.

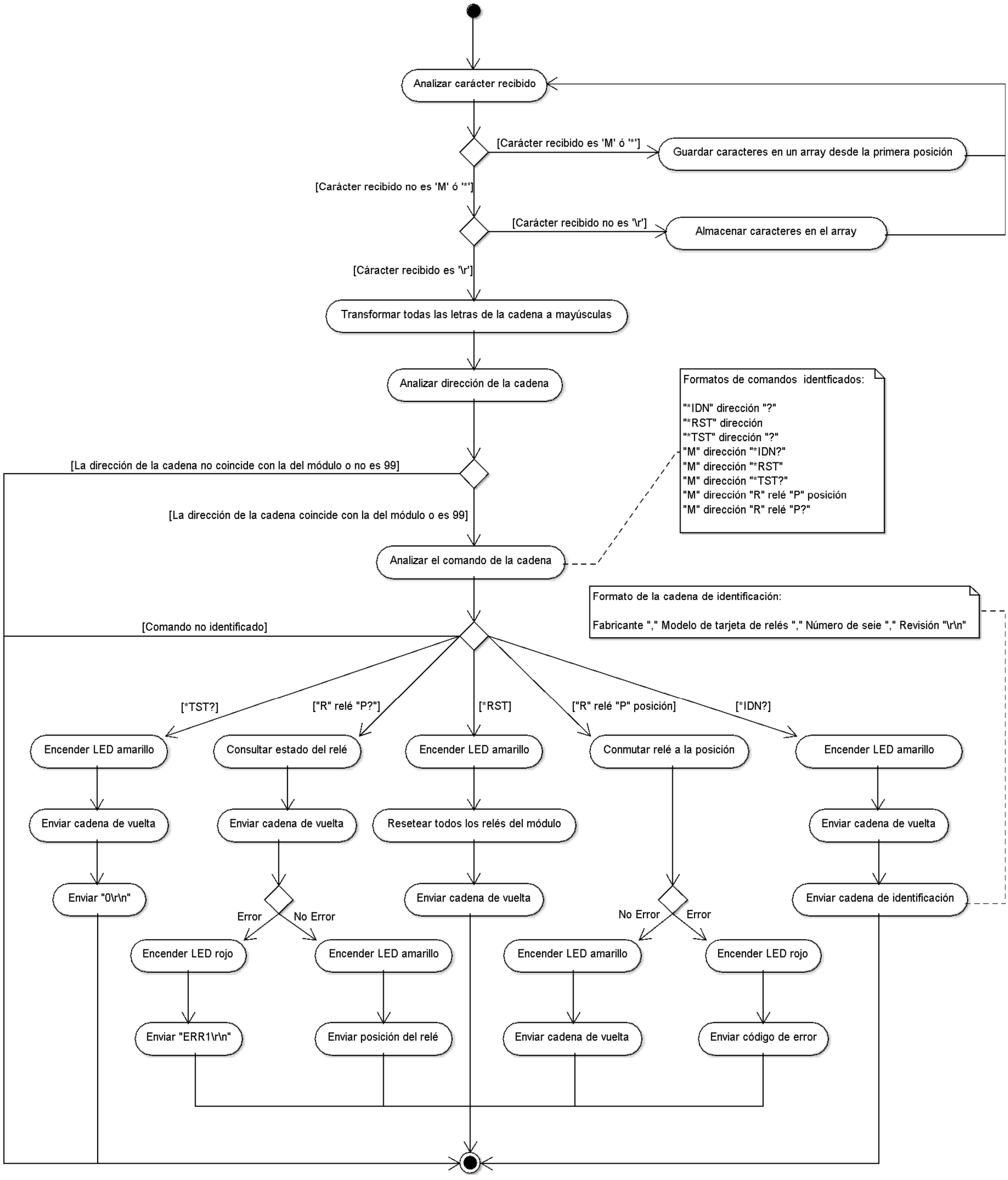


Figura 22. Diagrama de flujo del Procesamiento de la información

El algoritmo de procesamiento de la información consiste en iniciar el almacenamiento de caracteres al recibir "M" o "*" y finalizarlo al tratarse de "\r", de forma que se construye una cadena (*string*) que se analiza para tratar su contenido apropiadamente.

Tras transformar las minúsculas a mayúsculas por la comodidad que ofrece la indiferencia en la escritura de unas u otras durante el envío de caracteres, se examina la dirección a la que la instrucción recibida se refiere. En el supuesto de coincidir con la dirección del módulo o ser "99", la cadena alude al módulo en cuestión, es por ello que se continúa con el análisis de la cadena, los caracteres de los comandos empleados.

En el caso de que la cadena contenga un comando reconocible, el módulo señalado realiza las acciones pertinentes (operaciones atribuidas al comando y el encendido del LED amarillo o rojo dependiendo del resultado de estas) y envía un *ACK* (*Acknowledgement*), un mensaje que el destino de la comunicación transmite al origen para confirmar la recepción de la recepción, que consiste en devolver la misma cadena que se ha recibido.

Los comandos generales **IDN?* y **TST?* transmiten información acerca del módulo, tras mandar el *ACK* correspondiente. **IDN?* efectúa el envío de la identificación del módulo en cuanto a fabricante (*INDRA*), tipo de módulo de relés (catalogado gracias a las líneas de configuración en cada tarjeta), número de serie del módulo (registrado en el propio código del *firmware*) y revisión del *firmware* (constante grabada en el *firmware* tras cada actualización). El comando **TST?* se trata de una consulta acerca del estado del módulo. Muchos instrumentos *VXI* con funciones complejas realizan pruebas sobre ellos mismos (*self-test*) para determinar su disposición ante futuras operaciones; sin embargo, dada la naturaleza de la aplicación del sistema únicamente responde a la llamada con un *0* (unido a los caracteres "\r\n") que indica que se encuentra preparado para actuar.

El comando *MxxRyyPz* simplifica en una sola cadena el módulo (*xx*), el relé (*yy*) y la posición (*z*) a conmutar, por lo que incorpora todos los datos necesarios para comenzar la subrutina de conmutar el relé a la posición. Sin embargo, puesto que cada módulo de relé incorpora relés distintos, en cuanto a número de ellos y la estructura que adoptan (*SPDT* y *SPQT*), es necesario establecer una forma de relacionar cada posición de relé con las salidas del *microcontrolador* de control de los *drivers*.

Los requisitos del sistema imponen la elusión de posiciones intermedias en los relés, lo que presenta un obstáculo en los relés *SPQT* ya que emplean dos relés SPDT para su constitución y por consiguiente, una señal aplicada en una posición que debe llevarse a otra puede implicar su paso por otras posiciones, lo que no es permisible. Puesto que el tiempo en el que la varilla del relé se traslada de una posición a otra es mucho mayor que la velocidad de trabajo del microprocesador, basta con cambiar de posición ambos relés casi a la vez, con dos instrucciones consecuentes seguidas.

La solución a estos problemas radica en la utilización de un *array* de dos dimensiones, una dimensión que alberga el tipo de módulo de relé y otra que contiene los datos de configuración de relés de cada tipo de módulo. Los datos de configuración de los relés se establecen en estructuras, compuestas a su vez por estructuras, de manera que una gran estructura contiene tres estructuras:

- La estructura que indica el relé y la posición.
- La estructura que enlaza la posición del relé con la salida del *microcontrolador* (pin, puerto y estado) establecida en el diseño del *hardware*.
- La estructura que relaciona una segunda posición del relé, en caso de mantener una configuración de más de dos posiciones (*SPQT*) compuesta por dos relés dobles (añade otro pin, estado y puerto para su control).

La figura 23 muestra la estructura que almacena la información que vincula cada posición de relé con las salidas del *microcontrolador*.

```
typedef struct
{
   struct
   {
      uint16_t Relay;
      uint16_t Position;
   } Relay;

   struct
   {
      GPIO_TypeDef *Port;
      uint16_t Bit;
      uint16_t State;
   } Pin_1;

   struct
   {
      GPIO_TypeDef *Port;
      uint16_t Bit;
      uint16_t State;
   } Pin_2;
} RELAYS_CONFIG;
```

Figura 23. Estructura de los datos de configuración de los relés

Así, la búsqueda de la posición del relé en el *array* de cada tipo de módulo, determina los pines del *microcontrolador* y sus estados para realizar los cambios pertinentes.

La conmutación de un determinado relé a la posición queda reducida a su búsqueda en una lista (*array*). Primero se rastrea el relé, y una vez encontrado (en caso contrario se retorna un error) se busca la posición, y tras hallarla (en otro supuesto se envía otro error distinto) se conoce el estado de cada uno de los pines involucrados en la conmutación del relé y se guarda la nueva posición del relé en un *array*. A continuación se aplican los cambios en las salidas del *microcontrolador*, y si se trata de un relé con estructura *SPQT*, entonces se debe lanzar dos instrucciones seguidas acerca de las salidas del *microcontrolador*.

La figura 24 aclara la acción de conmutar un relé a una posición mediante un diagrama de actividades.

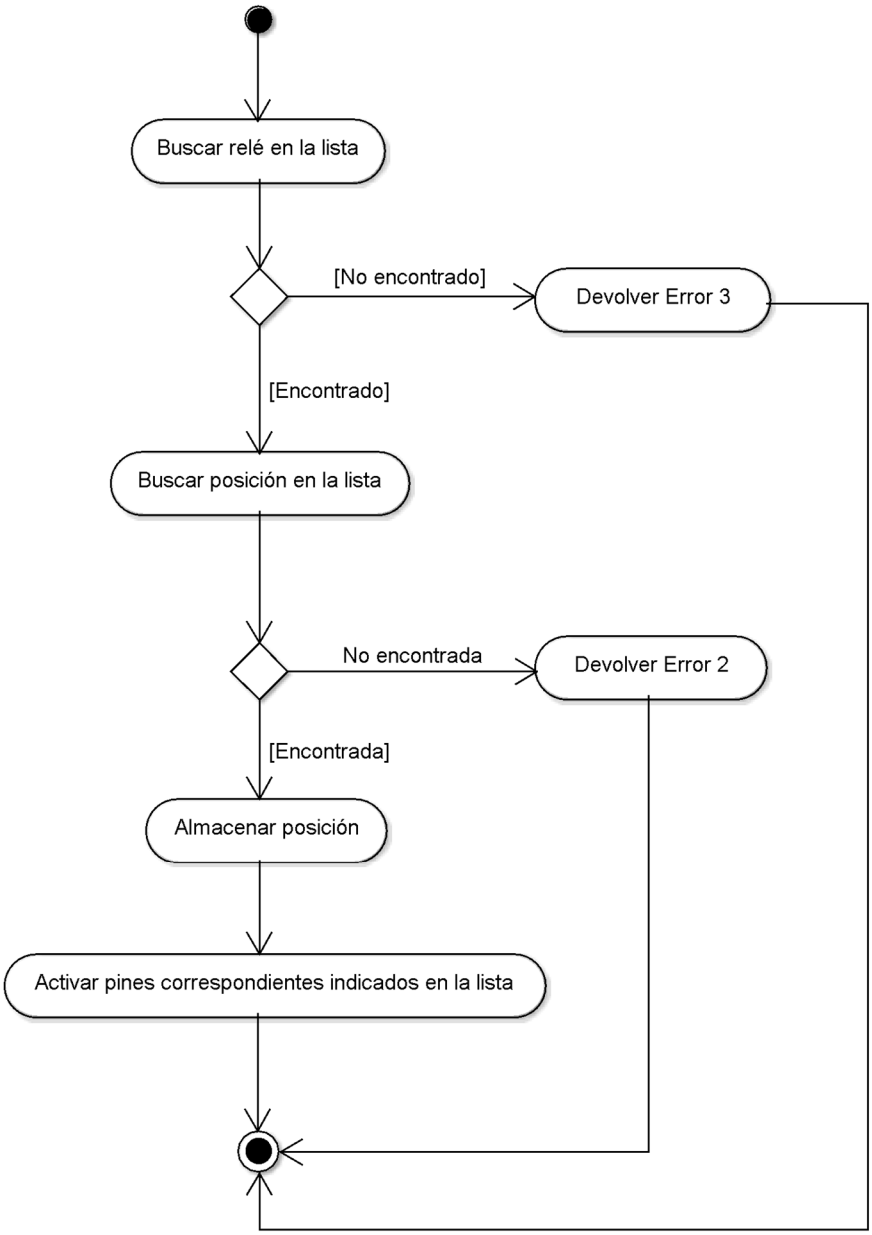


Figura 24. Diagrama de flujo de Conmutar relé a la posición

El comando *MxxRyyP?* expresa una consulta de la posición (estado) del relé. El algoritmo consiste en examinar el lugar correspondiente al relé indicado, en el *array* en el que se han almacenado las posiciones de los relés al conmutarlos. De manera que si el relé se encuentra en el *array* se envía la posición que mantiene, tras transmitir el *ACK*, y si no es localizado se retorna un error. En la figura 25 se ilustra el procedimiento en un diagrama de flujo.

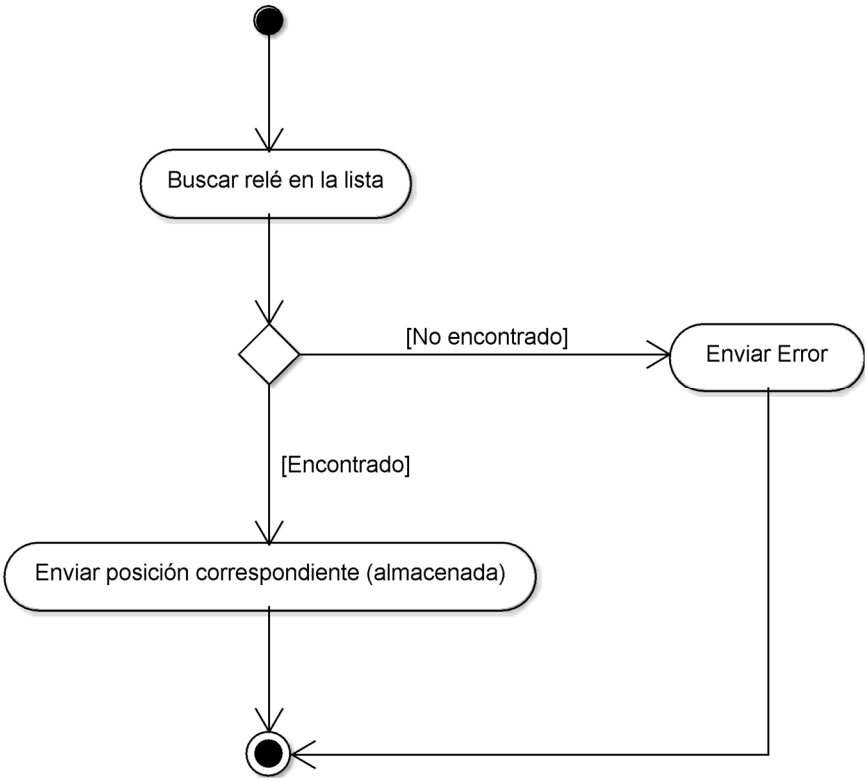


Figura 25. Diagrama de flujo de Consultar estado de relé

Por último, el comando **RST* efectúa una llamada a un reinicio de todos los relés, es decir, todos los relés del módulo deben volver a la posición inicial (*NC* o 0). Dado que la conmutación de todos los relés implica una llamada individual a cada uno de ellos, es conveniente emplear el algoritmo usado en la conmutación de un relé. Se comienza conmutando a la posición por defecto del primer relé y se continúa reiniciando el siguiente relé hasta que la subrutina de conmutar relé a la posición devuelva el error correspondiente a no haber encontrado el relé indicado. La figura 26 representa el método descrito a través de un diagrama de actividades.

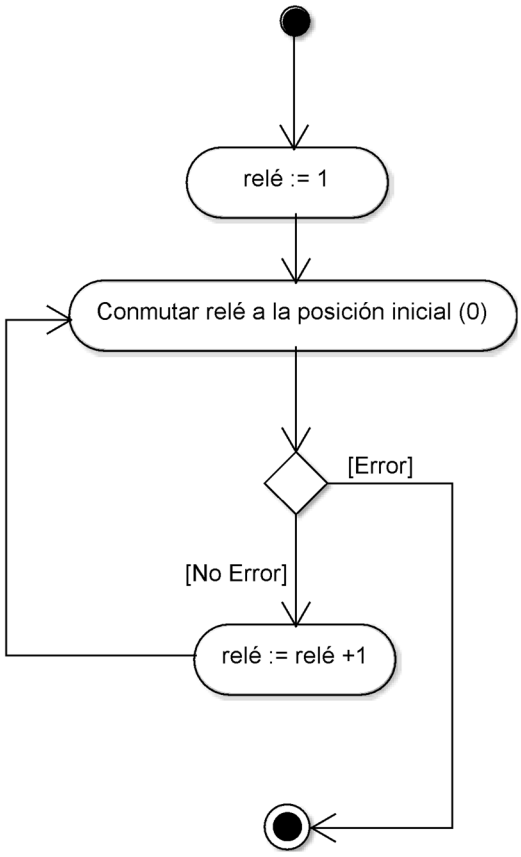


Figura 26. Diagrama de flujo de *Resetear* todos los relés del módulo

4.2.6 Depuración y descarga del *firmware*

Tras la implementación del código, es necesaria la compilación del código tanto para comprobar posibles errores como para su posterior interpretación por el *microcontrolador*. El entorno *µVision* incorpora un simulador (*Start/Stop Debug Sesión*) del microcontrolador, por lo que es posible realizar simulaciones con el propio *microcontrolador STM32F101V8* (como fue el caso) o sin él, examinando cualquier valor o variable, lo que permite una depuración exhaustiva sobre la funcionalidad.

Tras las pertinentes correcciones es conveniente observar que el tamaño del *firmware* no excede la capacidad del *microcontrolador*, tanto en memoria no volátil (*Code + RO Data + RW Data*) como en memoria volátil (*RWData + Zi Data*). Los valores calculados y ofrecidos por *µVision* distan bastante de las prestaciones otorgadas por el *microcontrolador*, 18220 bytes empleados en memoria no volátil frente a 64 Kbytes de *Flash* y 2512 bytes de memoria volátil lejos de los 10Kbytes de *SRAM.*

µVision crea automáticamente un fichero *HEX* (de acuerdo a los parámetros configurados), un fichero de texto que consiste en una serie de líneas consecutivas cuyo contenido se corresponde con las instrucciones que han de ser grabadas en la memoria (*Flash*) del *microcontrolador*; sin embargo, aunque la descarga del fichero HEX podría realizarse mediante multitud de programas dedicados a ello, *µVision* también incorpora un botón de descarga (*LOAD*) en los iconos del menú que se encarga de fijar el código máquina pertinente (*firmware*) en la memoria del *microcontrolador*.

Para comunicar el *microcontrolador* con el *PC* se emplea un depurador/programador para la familia de *microcontroladores STM32* (*ST-Link Debugger*) mediante varias interfaces (*JTAG/SWD*). [38]

4.3 *Driver*

El *driver* es el programa informático que interacciona entre el sistema operativo, en este caso *Windows*, y el *hardware*. Teniendo en cuenta que el sistema debe mostrar compatibilidad con el *bus VXI*, es ideal proyectar el *driver* en la misma arquitectura, *VISA* (*Virtual Instrument Software Architecture*).

Al tratarse de un *driver Plug&Play* la verificación del funcionamiento se torna más compleja, no obstante, gracias a *VISA Interactive Control* es sencillo comunicarse con el instrumento a través de la capa de *software* ofrecida por el *driver* y validar la respuesta del sistema.
La utilización del estándar *VISA* ofrece la posibilidad de desarrollar dos tipos de *drivers*:

- *Driver VXI Plug&Play*, un controlador de competencias limitadas, pero relativamente fácil de implementar.
- *Driver IVI*, un controlador de instrumentos más enrevesado y con atribuciones superiores.

El sistema de este proyecto presenta una interacción muy asequible con el controlador, es por ello que la alternativa de *VXI Plug&Play* es más que suficiente para la funcionalidad del caso propuesto.

NI-VISA es la implementación de *National Instruments* para el estándar *VISA*, incluye librerías de *software*, utilidades interactivas como *NI I/O Trace* (una herramienta para monitorizar las comunicaciones con el instrumento mientras la aplicación se ejecuta) y *VISA Interactive Control* (una utilidad que proporciona el acceso a toda la funcionalidad de *VISA* interactivamente, en un entorno gráfico fácil de manejar), y configuración de programas mediante *Measurement & Automation Explorer.* [20]

Dado que el lenguaje utilizado en el *driver* es C, otorga la posibilidad de elegir entre numerosos compiladores y depuradores, por ejemplo el propio entorno de desarrollo utilizado para crear la interfaz de usuario, ya que la validación de la *GUI* y el *driver* se encuentran estrechamente ligados.

Dada la diversidad de entornos de desarrollo de aplicaciones con los que es compatible (gracias a la librería de interfaz de *bus VXI* que se anexiona, el paquete *NI-VISA*), conviene evaluar las ventajas y los inconvenientes que otorga cada uno de ellos. Destacan:

- *LabVIEW*, es un entorno de desarrollo integrado creado por *National Instruments* especializado en informática industrial y científica. Utiliza un lenguaje gráfico para simplificar la implementación de programas informáticos complejos. Goza de numerosas librerías de funciones especialmente orientadas a la instrumentación, adquisición de datos, análisis matemático de medidas y la visualización. Así, se convierte en una herramienta especialmente útil para los sistemas de medida y ensayo.
- *LabWindows CVI*, es un entorno de desarrollo *ANSI C* integrado creado por *National Instruments* especializado a la realización de sistemas de control, instrumentación, prueba, simulación, medida y análisis matemáticos y *mecatrónicos*. Usa el lenguaje C, que unido a las bibliotecas que contiene ofrece las mismas posibilidades de *LabVIEW* con la ventaja añadida del lenguaje C. Es capaz de crear aplicaciones estables de alto rendimiento con herramientas de depuración e interfaz de usuario, que simplifica la elaboración de interfaces gráficas. [24]
- *TestStand*, es un motor de secuencias de instrucciones a la vez que un entorno de desarrollo para crear secuenciadores de pruebas según el producto a analizar. Inventado por *National Instruments* es la mejor salida para cargar secuencias de instrucciones de experimentación o acción con tolerancias y componentes para su comunicación con el *hardware*. [21]
- *Measurement Studio .NET*, es un conjunto de librerías y funciones dedicado al desarrollo veloz de soluciones industriales y de tecnologías *.NET* (explotables con *Visual Studio*). Esta infraestructura de *software* elaborada por *National Instruments* emplea el lenguaje C para posibilitar la construcción de funciones informáticas e instrumentación. [21]

- *Microsoft Visual C++*, es un entorno de desarrollo integrado para lenguajes de programación *C, C++* y *C++/CLI*. Engloba el desarrollo de aplicaciones construidas en la plataforma de *Windows*.
- Borland C++, es un entorno de programación de C y C++ para MS-DOS y *Windows*. Es el sucesor de *Turbo C++,* mejorándolo con un mejor depurador, el *Turbo Debugger*.
- *Microsoft Visual Basic*, es un entorno de desarrollo para un lenguaje de programación dirigido por eventos, *Visual Basic*. Integra un editor de textos para edición del código fuente, un depurador, un compilador, un enlazador y un editor de interfaces gráficas.
- *ATEasy*, es un entorno de desarrollo de aplicaciones y de pruebas para test funcionales, adquisición de datos, *ATE*, control de procesos y sistemas de instrumentación. Provee todas las herramientas necesarias para el desarrollo, la implementación y mantenimiento de los componentes de *software*. Está dirigido al soporte y la simplificación de aplicaciones de sistemas *ATE*, con ciclos de vida del producto duraderos. [57]

VISA ofrece interfaces de programación entre el dispositivo y el desarrollo de entornos como *LabVIEW, LabWindows/CVI* y *Measurement Studio for Microsoft Visual Studio* [20]. Dado que el código proporcionado por el *driver* es lenguaje *ANSI C*, la elección más favorable es *LabWindows/CVI*, un entorno de desarrollo para crear aplicaciones estables de alto rendimiento en la industria militar, aeroespacial, de manufactura, de telecomunicaciones y automotriz. Promueve la eficiencia del desarrollo con asistentes de configuración de *hardware*, herramientas de depuración, utilidades de interfaz de usuario y bibliotecas de medidas y análisis.

4.3.1 Funcionalidad

El *driver* constituye la capa de *software* intermedia encargada de conformar una comunicación entre la capa de la interfaz gráfica de usuario y la capa de *firmware*. La manera de realizar esta comunicación determina la naturaleza del instrumento *VXI* (extendidos, de memoria, basados en registros y basados en mensajes) según las especificaciones (ver anexo A.1.2.5).

Aunque el diseño del *firmware* encuadra el sistema en una clase de instrumento *VXI*, el controlador de los módulos de relés remata la definición de instrumento basado en mensajes. Los dispositivos basados en mensajes poseen la capacidad de ofrecer servicios más complejos y diversos a pesar un coste ligeramente más elevado, pero las prestaciones otorgadas por el *microcontrolador* permite el aprovechamiento de una comunicación basada en mensajes sin incrementar el precio.

La interacción humana-instrumento es posible a través de la interfaz de usuario o un conjunto de comandos de software accesibles, pero ambos medios deben ser implementados por el controlador en una arquitectura de software de instrumentos virtuales (*VISA*). El driver debe proveer un grupo de funciones en una sintaxis propia de un lenguaje estructurado (*C*) con compatibilidad con *VXI* (*VISA*), que expriman la funcionalidad del dispositivo y resulten

intuitivas a la hora de traducir la información proveniente de aplicaciones con un lenguaje del más alto nivel a unos comandos interpretables por el *firmware* del *microcontrolador*.

4.3.2 Comunicación y tratamiento de los datos

El driver implementa una serie de funciones que permiten la totalidad del control de cada módulo de relés, de forma que con o sin el uso de la interfaz gráfica se gobierna la comunicación con la capa del *firmware* (llamada desde programas de test).

El requisito de afinidad con *VXI* implica el uso de *VISA*, un modelo de software y una *API* específicos. *VISA* determina una arquitectura que consiste en numerosos recursos, que encapsulan la funcionalidad del dispositivo y ofrecen servicios especializados a aplicaciones u otros recursos.

VISA Resource Manager es un recurso como cualquier otro en el sistema, que otorga conectividad a todos los recursos registrados en él (interfaz *Ethernet*), de forma que permite el gobierno y acceso individuales de los recursos (abrir, cerrar y encontrar recursos, establecer atributos, generar eventos, etc.). [54]

VISA Instrument Control Resource es otro recurso que define las operaciones básicas y atributos de la plantilla de recursos *VISA*, pero cada recurso añade operaciones específicas y atributos que le permiten realizar la tarea para la que han sido especialmente diseñados (abrir y cerrar relés). [54]

Las funciones básicas y específicas del controlador, ajustándose a *VISA*, cubren todas las necesidades de la aplicación:

- *ViStatus _VI_FUNC indrasmip_init(ViRsrc resource, ViPSession vi)*. Se encarga de establecer la comunicación con el recurso indicado (*resource*), en este caso, la interfaz *Ethernet-Serie*. Primero crea una sesión única con *VISA Resource Manager* (*viGetDefaultRM(&resMgrSession)*) y a continuación conforma una conexión con el instrumento deseado (*viOpen(resMgrSession, resource, VI_NULL, VI_NULL, vi)*), que resulta en una nueva y exclusiva sesión *VISA* (*vi*).
- *ViStatus _VI_FUNC indrasmip_close(ViSession vi)*. Se ocupa de desconectar la comunicación de la sesión (vi) y concluir el sistema *VISA*.
- *ViStatus _VI_FUNC indrasmip_reset(ViSession vi, ViInt16 addr)*. Envía el comando **RST* al módulo señalado (*addr*) una vez creada la sesión *VISA*.
- *ViStatus _VI_FUNC indrasmip_selftest(ViSession vi, ViInt16 addr, ViChar test[])*. Transmite el commando **TST?* al módulo de relés seleccionado (*addr*), en la sesión *VISA* proporcionada (*vi*), y almacena el resultado del test en un *array* (*test*).
- *ViStatus _VI_FUNC indrasmip_error_query(ViPSession vi, ViInt16 addr, ViPInt32 errorNumber, ViChar errorString[])*. Dado que no se almacenan los errores debido a la necesidad del uso de registros en la memoria del *microcontrolador*, esta función goza

de poca utilidad; sin embargo, verifica la existencia de una sesión *VISA* válida (*vi*) y acomoda una posible futura implementación de conservación de errores.

- *ViStatus _VI_FUNC indrasmip_error_message(ViStatus errorCode, ViChar errorMessage[])*. Representa una función que es llamada cuando la *GUI* detecta algún error, entonces le suministra a esta función el código de error (*errorCode*), y esta le retorna el mensaje correspondiente al error en un *array* (*errorMessage*).
- *ViStatus _VI_FUNC indrasmip_idn(ViSession vi, ViInt16 addr, ViChar manufacturer[], SMIP_Model *model, ViChar serial_number[], ViChar firmware_rev[])*. Se trata de una solicitud de identificación al módulo de relés (*addr*) en la sesión *VISA* establecida (*vi*), en el que este responde su identidad (fabricante, tipo de módulo de relés, número de serie y revisión firmware) en una única cadena, que esta función se encarga de descomponer y almacenar en cuatro campos (*manufacturer, model, serial_number* y *firmware_rev)* independientes de acuerdo a la información a la que aluden.
- *ViStatus _VI_FUNC indrasmip_set_relay_list(ViSession vi, ViInt16 addr, ViInt16 relay_list[], ViInt16 relay_pos[], ViInt16 rly_cnt)*. Consiste en la conmutación de una lista de relés (*relay_list)*, integrada por un número de relés determinado (*rly_cnt*), del módulo indicado (*addr)* a las posiciones señaladas en una lista (*relay_pos)* a través del recurso con el que se ha instaurado la sesión *VISA* (*vi*), para lo que envía los comandos de conmutación (*MxxRyyPz*) correspondientes uno por uno.
- *ViStatus _VI_FUNC indrasmip_QueryRelayPosition(ViSession vi, ViInt16 addr, ViInt16 relayNmbr, ViInt16 *Pos)*. Demanda la posición (*Pos*) en la que se encuentra el relé (*relayNmbr)* del módulo determinado (*addr*), de la interfaz Ethernet-Serie con la que se ha instaurado una sesión *VISA* (*vi*), mediante el envío del comando *MxxRyyP?*.

Todas estas funciones comparten la misma naturaleza de retorno, *ViStatus* ofrece el resultado de la operación de cada función de manera que la *GUI* conozca los detalles de la finalización; es más, si durante el transcurso de la función se recibe algún error, ya sea por parte del *firmware*, como en la comprobación de *ACKs,* elaborada por una pequeña función dedicada a tal propósito, o de origen desconocido, la función se aborta y devuelve de inmediato el código de error, para que sea conocido y procesado por la *GUI* (llamada a la función *indrasmip_error_message(…)* y su tratamiento como error en una ventana nueva). [49]

_VI_FUNC es un prefijo propio de los controladores *VXIPlug&Play,* que constituye una macro que hace referencia al tipo de llamada dependiendo del sistema en que se encuentra.

4.4 Interfaz gráfica de usuario

La interfaz gráfica de usuario (*GUI*) es un programa informático dedicado a una comunicación abierta entre el usuario y un dispositivo electrónico a través de gráficos e indicadores visuales, que muestran la información y las acciones disponibles. Se elabora en el entorno de desarrollo *LabWindows/CVI*, junto con el *driver*.

El entorno gráfico facilita una comunicación fluida entre usuario e instrumento y concentra la atención del usuario en los elementos importantes de la manipulación de relés.

El desarrollo de la composición gráfica y comportamiento de la interfaz es una parte importante de la aplicación de programación de *software* en el área de la interacción hombre-máquina. Existen numerosos principios fundamentales que definen la consecución de una buena *GUI* [17] [44]:

- Anticipación, la interfaz gráfica de usuario debe adelantarse a las necesidades de este, mostrando toda la información y herramientas indispensables para su trabajo.
- Uso valores por defecto, que deben tener sentido y ayudar al usuario en la elección.
- Reducción de la latencia, eliminando de la aplicación cualquier elemento que no esté ayudando.
- Prevención de periodos de aprendizaje, potenciando la facilidad de uso y la usabilidad, de forma que la aplicación sea intuitiva (con iconos, dibujos, sin sorpresas…), incremente la sensación del control del usuario sobre el sistema y favorezca la navegación (todas las posibles acciones tienen que ser accesibles desde el menú principal dentro de lo posible).
- Acromatismo, es recomendable eludir colores chillones o con semejanzas con el fondo.
- Disminución de la necesidad de memorización del usuario, usando controles gráficos y limitando la carga de información.
- Autonomía, es conveniente mantener informado al usuario del estado del sistema de forma actualizada y cómodamente visible.
- Maximización de la eficacia del usuario, es indicado *bus*car la productividad del usuario en lugar de la del ordenador, mantener ocupado al usuario (sin esperas), mejorar la arquitectura de la aplicación (diseño gráfico), proporcionar mensajes de ayuda claros y concisos al usuario, etiquetas con palabras significantes (o aún mejor que comiencen con ellas), etc.
- Reducción de la posibilidad de cometer errores, usando las restricciones de valores introducidos, inhibiendo acciones no permitidas o librándose de controles físicamente difíciles de manejar. En caso de que se cometa un error, suministrar una información entendible por el usuario y facilitar la recuperación del estado anterior.
- Realimentación, la interfaz de usuario debe comprometerse a otorgar una respuesta inmediatamente a cualquier acción del usuario.

4.4.1 Funcionalidades de la interfaz

La funcionalidad de la interfaz gira en torno a las competencias de los relés y la información pertinentes del instrumento que los alberga.

Existen tres funciones principales relativas a los relés:

- Resetear todos los relés de un módulo de relé, que desempeña la labor de conmutar todos los relés del módulo elegido a la posición inicial, y por tanto por defecto.

- Visualizar el estado de un relé, que se encarga de retratar la posición del relé seleccionado.
- Conmutar un relé, que ejerce la conmutación de un relé a la posición escogida.

Aunque suponen tres acciones independientes, la visualización de un relé y su conmutación se encuentran ligeramente ligadas, ya que el esquema del relé consta de una línea gruesa que conecta dos posiciones (común, *NC*, y cualquier otra), las cuales constituyen controladores que al ser pulsados realizan la conmutación del relé a dicha posición.

Si bien los relés se encuentran conformando estructuras más complejas (*SPQT*), como en el módulo de relés *IND5003*, se representa la figura del relé con cuatro posibles posiciones y una posición común, de manera que aunque se puedan conmutar dos relés, funcionalmente simboliza únicamente la conmutación de un relé de cuatro posiciones.

Además, la interfaz gráfica también otorga información sobre el módulo de relés y la interfaz *Ethernet-Serie*, tanto en elementos dedicados a ello como en los propios controladores. Aunque no conforma funcionalidad adicional, se notifican datos que facilitan de cara al usuario los cometidos fundamentales de la interfaz gráfica:

- Módulo de relé elegido.
- Relé seleccionado.
- Dirección *IP* (Recurso *IP*) de la interfaz Ethernet-Serie.
- Tipo de módulo de relé.
- Tipo de relé.
- Información explicativa ante la aparición de un error.
- Posiciones de los relés y su correspondencia en los pines del conector.

La información de la correlación en terminales del conector se obtiene de un fichero *.csv* (*Comma Separated Value*), en la que los datos de cada posición de relé se agrupan en una línea separando cada cadena de texto independiente mediante puntos y coma. Este tipo de ficheros también son visibles y editables por *Excel* (el punto y coma tras cada conjunto de caracteres se presenta como una celda nueva), por lo que supone una buena opción a la hora de modificar la relación de pines y posiciones.

La figura 27 muestra el diagrama de casos de uso de la interfaz gráfica.

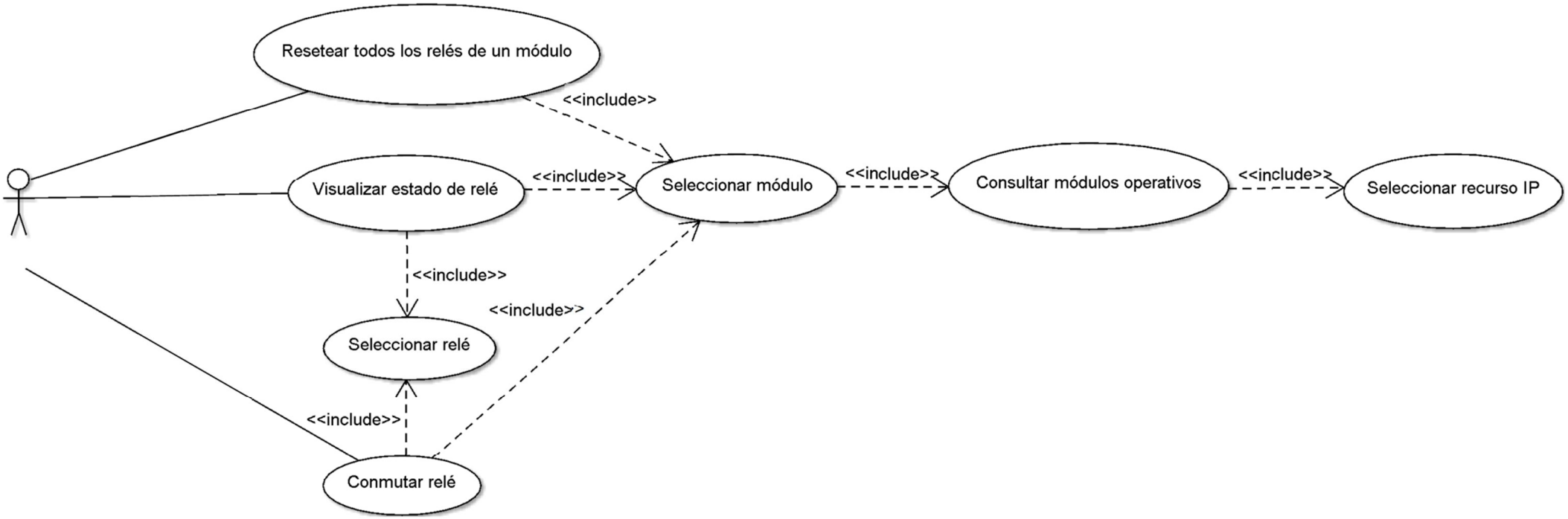


Figura 27. Diagrama de casos de uso de la interfaz

4.4.2 Elementos de la interfaz

Los elementos de la interfaz gráfica contribuyen a que el control y la visualización de la información resulte una tarea sencilla, útil y cómoda. La *GUI* distribuye su funcionalidad en tres ventanas:

- Un primer panel (*Relays Initializer*), inicial, que otorga la posibilidad de iniciar la comunicación con el recurso *IP* de la interfaz *Ethernet-Serie*, por lo que desaparece una vez que ha cumplido su cometido.
- Un segundo panel (*Relays Controller*), principal, que ofrece la mayoría de elementos básicos y el control total sobre los módulos de relés conectados a la interfaz *Ethernet-Serie*.
- Un panel dedicado a la visualización de errores (*Error*), que en el supuesto de que se produzca algún error, suministra información acerca del error que se ha producido.

El panel inicial constituye una ventana pequeña que aparece al ejecutar la aplicación, tal y como ilustra la figura 28.

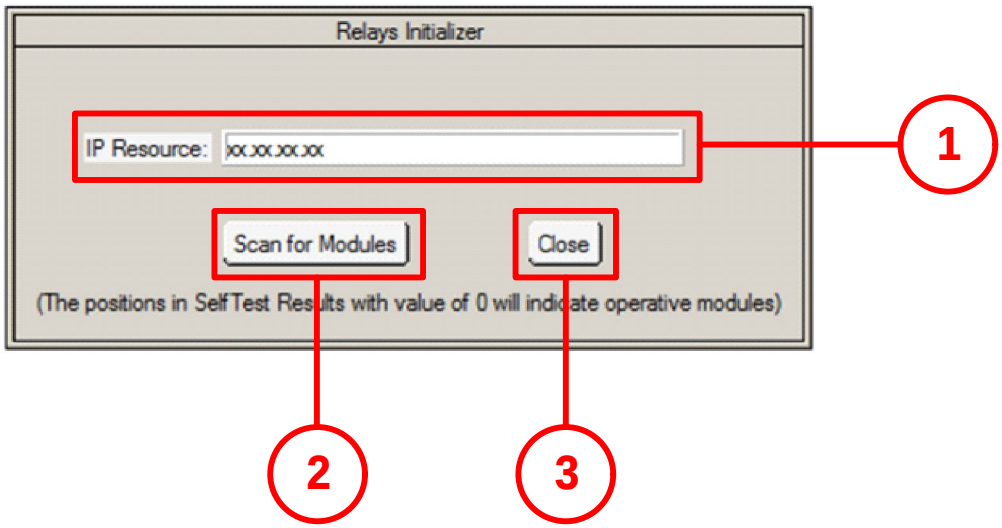


Figura 28. Panel inicial de la interfaz gráfica

El panel *Relays Initializer*, mostrado en la figura anterior, se encuentra compuesta de tres elementos:

1. Una caja de texto editable con valor inicial xx.xx.xx.xx que permite al usuario introducir la dirección IP (*IP Resource*) de la interfaz *Ethernet-Serie*.
2. Un botón de comando (*Scan for Modules*), cuyo accionamiento produce el análisis de la operatividad de los posibles módulos conectados a la interfaz *Ethernet-Serie*.
3. Un botón de comando (*Close*), que presionado finaliza la aplicación con el consiguiente cierre de ventanas.

El panel principal se establece en una ventana más grande, como se observa en la figura 29, y conforma el núcleo de la interfaz gráfica.

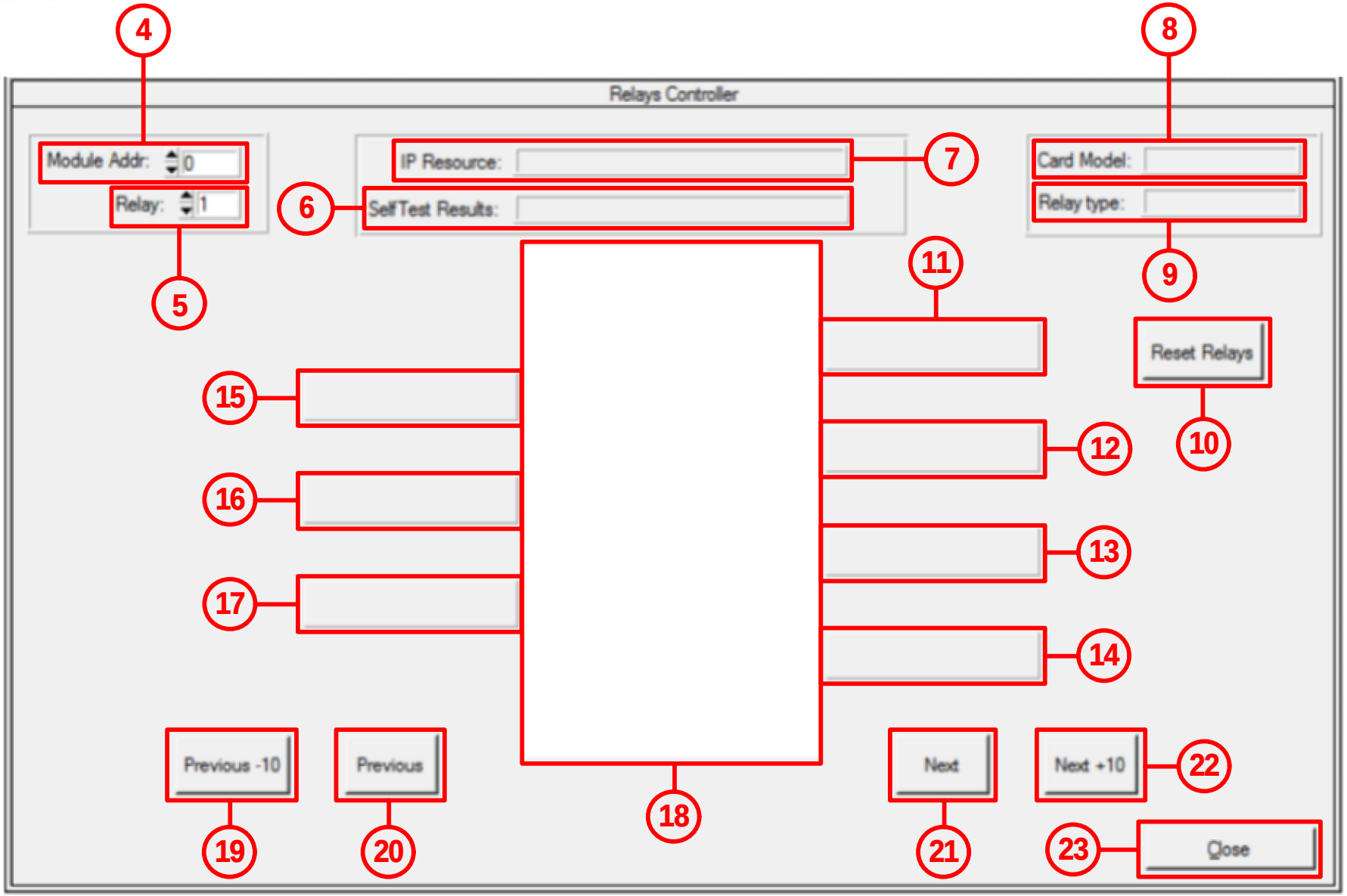


Figura 29. Panel principal de la interfaz gráfica

El panel *Relays Controller* que ilustra la figura anterior alberga prácticamente toda la funcionalidad del dispositivo, que suministra a través de diversos elementos:

4. Un control numérico que establece la dirección del módulo (*Module Addr*) con el que se comunica la interfaz, editable bien mediante la inserción de un número o en su lugar a través del uso de flechas. Inicialmente posee el valor por defecto de la dirección del módulo de menor valor y solo es posible ajustar su valor a una de la direcciones de módulos operativos, analizados previamente y mostrados mediante un *0* en el elemento 6 (*SelfTest Results*).
5. Un control numérico que selecciona el número de relé (*Relay*) a ser manipulado, modificable por medio de flechas o valores introducidos por teclado. En un principio adopta el valor 1 y se encuentra restringido superiormente por el número máximo de relé que tiene cada tipo de módulo.
6. Una caja de texto que muestra los resultados del *Self Test* (*SelfTest Results*) de la interfaz *Ethernet-Serie*, es decir, examina los módulos y representa el estado de cada uno de ellos (desde el módulo con dirección 0 hasta el módulo con dirección 31)) en forma de texto, de manera que su posición en la cadena corresponde a su dirección y un 0 indica su operatividad, mientras que un 1 señala su indisponibilidad.
7. Una caja de texto que expone la dirección IP (*IP Resource*) de la interfaz *Ethernet-Serie* con la que se ha establecido una conexión mediante el panel inicial.
8. Una caja de texto que indica el tipo de módulo de relés (*Card Model*) elegido a través del elemento 4.
9. Una caja de texto que señala la configuración que conforman los relés (*Relay Type*), o sea, en el caso de este sistema *SPDT* o *SP4T* (*SPQT*).
10. Un botón de comando (*Reset Relays*) que tras ser pulsado realiza la función de resetear todos los relés del módulo seleccionado en el elemento 4.
11. Un mensaje de texto que muestra la posición del relé *NC* o *NC1,* posición inicial o por defecto, y su correspondencia en el terminal del conector. Además, el evento de clic dentro de su contorno activa la conmutación del relé escogido en el elemento 5 a esta posición.
12. Un mensaje de texto que expone la posición del relé *NO1*, en el caso de la configuración de relés *SPQT*, o *NO*, en el supuesto de la conformación de relés *SPDT* o *DPDT* (inexistente en este sistema pero desarrollado de todas formas), y su correlación en el pin del conector. También realiza la conmutación del relé elegido en el elemento 5 a esta posición si se pulsa sobre él.
13. Un mensaje de texto que muestra la posición del relé *NO2*, en las estructuras *SPQT*, o *NC2*, en las configuraciones *DPDT*, y su relación con el pin del conector; en el caso de tratarse de relés en una disposición *SPDT* se atenúa, volviéndose invisible e inutilizable. Controla la conmutación del relé del elemento 4 a esta posición, de modo que un cliqueo encima activa la acción.
14. Un mensaje de texto que indica la posición del relé *NO3*, en las configuraciones *SPQT*, o *NO2* en las conformaciones *DPDT,* y su correlación con el terminal del conector; en las estructuras *SPDT* se inhabilita y se vuelve imperceptible. Gobierna la conmutación del relé a esta posición, de forma que cliquear en la caja de texto supone conmutar el relé a la posición.

15. Un mensaje de texto que se habilita en las configuraciones *SPDT* y *DPDT,* y muestra la posición común *COM* y su relación con el pin del conector.
16. Un mensaje de texto que se activa únicamente en estructuras *SPQT* y expone la posición común *COM* y su correspondencia con el terminal del conector.
17. Un mensaje de texto, visible exclusivamente en las conformaciones *DPDT*, que señala la posición común *COM* y su correlación con el pin del conector.
18. Un lienzo que representa la figura del relé elegido en el elemento 5, de acuerdo a su posición y configuración (*SPDT, SPQT* o *DPDT*). Cualquier acción sobre el relé, ya sea la conmutación, cambio de módulo de relés, cambio de relé, reseteo de todos los relés… actualiza la visualización del relé, por lo que siempre muestra la conexión que se establece entre terminales del conector.
19. Un comando de botón (*Previous*), cuyo accionamiento disminuye el valor del elemento 5 en una unidad, con las consecuencias que ello supone en el resto de elementos. En el caso de no poder descontar del valor del elemento 5 (el valor mínimo es 1) se inhabilita desde ese momento.
20. Un comando de botón (*Previous -10*), cuya activación desciende el valor del elemento 5 en 10 unidades, con las correspondientes actualizaciones en los elementos de la interfaz gráfica. En el supuesto de no poder disminuir hasta un valor 10 unidades por debajo, se atenúa e inutiliza desde ese instante.
21. Un comando de botón (*Next*) que al ser presionado aumenta el valor del elemento 5 en una unidad, con las consecuentes repercusiones en algunos de los elementos de la interfaz gráfica. Si el elemento 5 se encuentra ya en su valor límite entonces se desactiva el botón automáticamente.
22. Un comando de botón (*Next +10*) que añade 10 unidades al valor del elemento 5 si es pulsado, con su debida trascendencia en los elementos de la interfaz gráfica. En el caso de que este aumento supere el valor máximo establecido por el elemento 5, entonces se inhabilita el botón.
23. Un comando de botón (*Close*) que de ser presionado concluye la aplicación, y por consiguiente todas sus ventanas.

El panel de errores se acomoda en una ventana pequeña, de suficiente anchura como para mostrar cualquier mensaje de error, tal y como se aprecia en la figura 30.

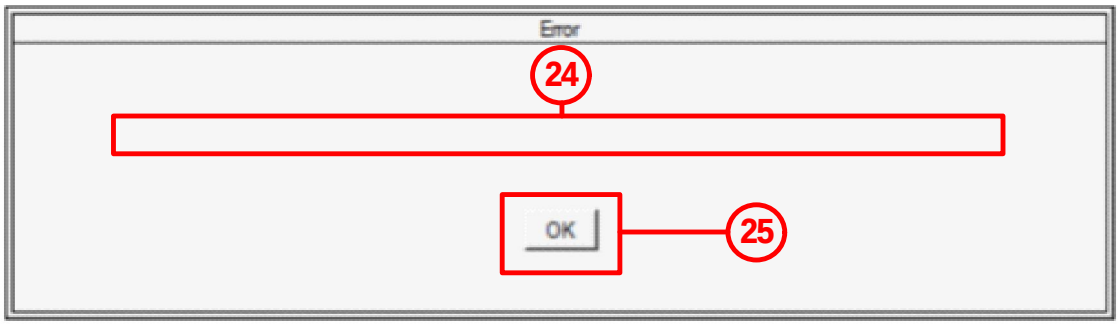


Figura 30. Panel de errores de la interfaz gráfica

El panel *Error*, dispone únicamente de dos elementos que logran sobradamente el cometido de esta ventana:

24. Un mensaje de texto que notifica cualquier error que se haya producido durante la ejecución en cualquiera de las capas de software, con el valor del error y una pequeña descripción acerca del error proporcionada por el *driver*.
25. Un botón de comando que al ser accionado cierra la ventana de error, dando por sentado que el usuario ha sido advertido del problema.

4.4.3 Estilo de la interfaz

De acuerdo a una rigurosa metodología de diseño de la interfaz gráfica, se ha implementado una *GUI* distribuida en tres ventanas o paneles, de forma que en cada momento se disponga únicamente de las acciones disponibles y operables.

El panel de inicio se emplea exclusivamente para realizar un análisis de módulos en la interfaz seleccionada, por lo que el elemento para llevar a cabo dicho cometido se encuentra centrado ocupando la mayor parte de la ventana, así como el botón que desencadena esta acción.

El panel principal se halla dispuesto de tal forma que la visualización del relé y su información (posiciones y terminales) se localizan en el centro de la ventana, y puesto que la conmutación de relés constituye una función principal de la interfaz gráfica, coexiste y se complementa con esta representación del relé para lograr una mayor sencillez y comodidad hacia el usuario. Otra operación fundamental de la interfaz consiste en el reinicio de todos los relés de un módulo, por lo que se sitúa completamente visible y cuasi centrado.

Si bien son necesarios los controles y los datos, que determinan el módulo y el relé, para una manipulación propicia de acuerdo a la intención del usuario, se encuentran en la parte superior del panel principal, de forma que sea la información que primero perciba el usuario; y por el contrario los controles complementarios y de finalización de la aplicación se posicionan en la parte inferior. Aunque existen botones para cerrar la aplicación, la interfaz gráfica también proporciona, como cualquier ventana de los sistemas operativos *Windows*, iconos de control para minimizar, maximizar o cerrar la aplicación, que de acuerdo a la versión del sistema operativo se presenta con un estilo u otro.

El panel de errores cumple un simple propósito que consiste en una breve reseña del error acontecido, por lo que esta descripción se ubica en el centro de la ventana propiciando una buena visión del mismo, seguido verticalmente del botón que cierra esta ventana.

La disposición de todos estos elementos conforme a la metodología empleada en la *GUI* conforma un estilo de interfaz gráfica que facilita el manejo, otorga una respuesta veloz, ofrece flexibilidad a futuras implementaciones, detalla la información e interés y brinda un programa plenamente visible.

4.5 Integración y solapamiento de capas de *software*

La finalización de cada una de las capas permite la constitución de una estructura que permita el control de los módulos de relés de manera independiente, bien a través de la interfaz de usuario o en su lugar mediante algún sistema dedicado a la ejecución de programas de test (como *PAWS*) que emplee las funciones proporcionadas por el driver.

La figura 31 ilustra las capas de software empleadas en el sistema y las relaciones entre ellas, desde la perspectiva del usuario, de modo que el usuario solo interacciona con la *GUI* para gobernar los relés.

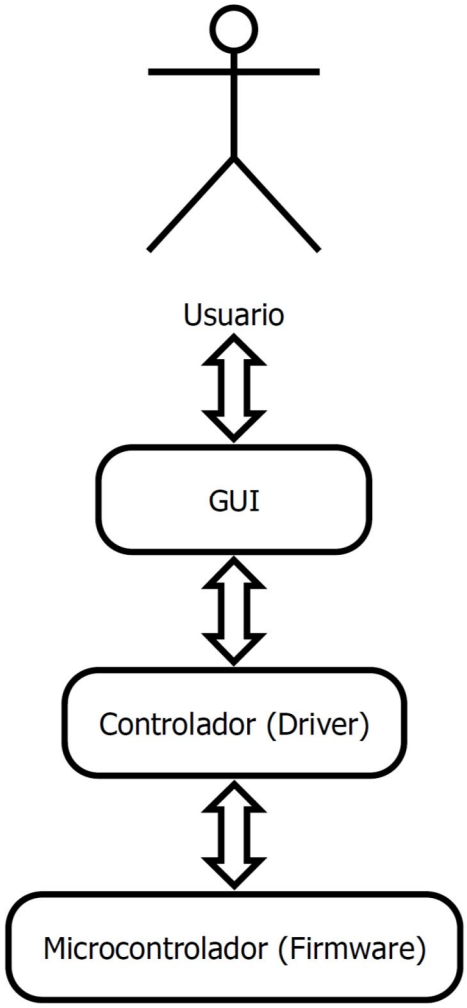


Figura 31. Jerarquía de capas de *software*

Desde un comienzo, el usuario parte de una situación en la que la interfaz *Ethernet-Serie* se encuentra conectado con un cableado Ethernet cruzado a un ordenador con sistema operativo

Windows. El usuario ejecuta el programa de la interfaz gráfica de usuario (*IndraSmip.exe*) para iniciar el control de los módulos de relés conectados a la interfaz *Ethernet-Serie*.

La *GUI* presenta, en un principio, un panel que le suministra al usuario un medio de introducir el recurso *IP* (dirección) correspondiente a la interfaz *Ethernet-Serie*, como muestra la figura 32.

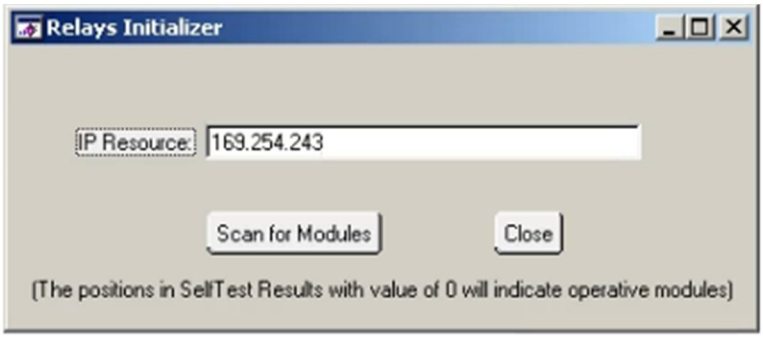


Figura 32. Panel de inicio de la *GUI*

El botón de "*Scan for Modules*" inicia una búsqueda de módulos de relés operativos que resulta en una comunicación entre las capas de software tal y como presenta la figura 33. Comienza con la solicitud del usuario, que es procesada por la interfaz y emite las correspondientes peticiones a la capa del *driver*, y esta a su vez se comunica a través del estándar *Ethernet* (ver anexo A.2.2) con la interfaz *Ethernet-Serie.* La interfaz *Ethernet-Serie*, como su propio nombre indica, transforma los paquetes de datos a *RS-232*, interpretable por el *microcontrolador* de cada uno de los módulos de relés.

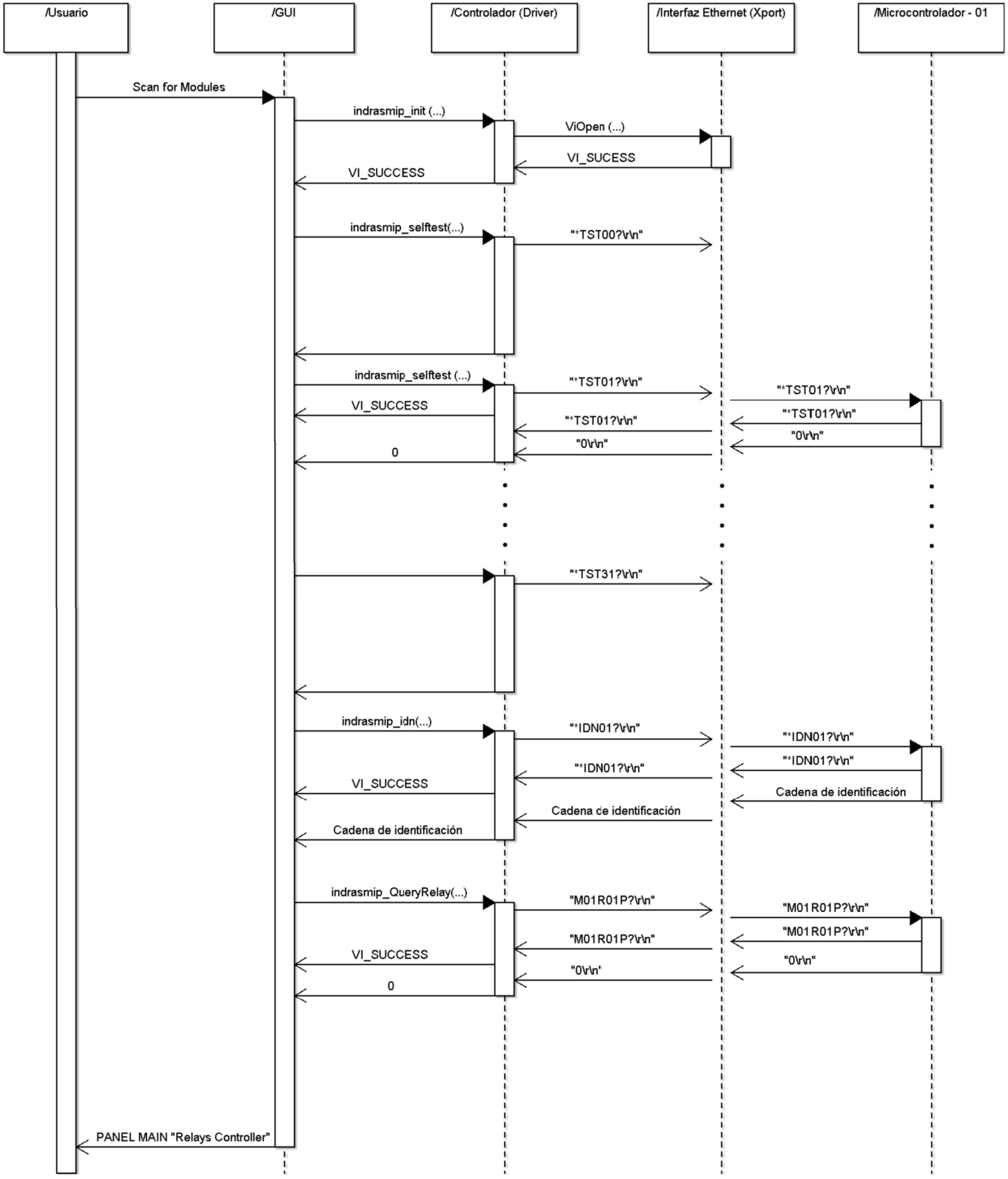


Figura 33. Diagrama de secuencia *software* – Inicio

Como se observa en la figura anterior, la solicitud "*Scan for Modules*" emprende la función *indrasmip_init(…)* en el *driver,* y este a su vez establece una conexión con la interfaz *Ethernet-Serie* por medio de la función *ViOpen(…),* que retorna el valor *VI_SUCCESS* si se ha completado con éxito hasta la interfaz de usuario. A continuación la *GUI* llama a la función *indrasmip_selftest(..)* del *driver,* encargada de comprobar el estado del módulo de relé por medio del comando **TST?,* para todos los módulos del sistema, uno por uno hasta alcanzar el último (módulo de relés con la dirección 31). Cada módulo de relés operativo responde además del consecuente *ACK*, a través de las capas de software, un 0 a la *GUI*, que interpreta su disponibilidad y la almacena.

De todos los módulos operativos, establece como módulo de relés por defecto aquel con menor dirección y le insta a identificarse, por medio de la función *indrasmip_idn(…)* del controlador, que utiliza el comando **IDN?* hacia el *firmware.* Tras recibir el *ACK*, el driver lo interpreta como una consecución del procedimiento adecuada (transmite *VI_SUCCESS* a la GUI), y la interfaz gráfica de usuario procesa y guarda la información recibida acerca del fabricante, tipo de módulo de relés, número de serie y revisión de firmware.

Por último, la GUI llama a la función *indrasmip_QueryRelay(..)* del controlador, que examina la posición actual del primer relé del módulo, enviando el comando *MxxR01P?*. El módulo de relés en cuestión (*xx*), que en el caso ilustrado por la figura se trata del módulo de relés con la dirección 01, responde el *ACK* y luego la posición en la que se encuentra el primer relé (concretamente en el supuesto de la figura en la posición inicial, 0), que es transferida hasta la *GUI.*

La interfaz gráfica de usuario ya posee todos los datos necesarios como para presentar el panel principal, por lo que transforma el formato de los datos a uno más sencillo de visualizar por el usuario, como se aprecia en la figura 34. Se dibuja el relé de acuerdo a su disposición con sus respectivas correspondencias de posiciones y pines, se muestra la información del recurso *IP* (*IP Resource*), los módulos operativos a través de ceros (*SelfTest Results*), el tipo de módulo de relés (*Card Model*), la clase de relé y los controles que permiten gobernar los relés.

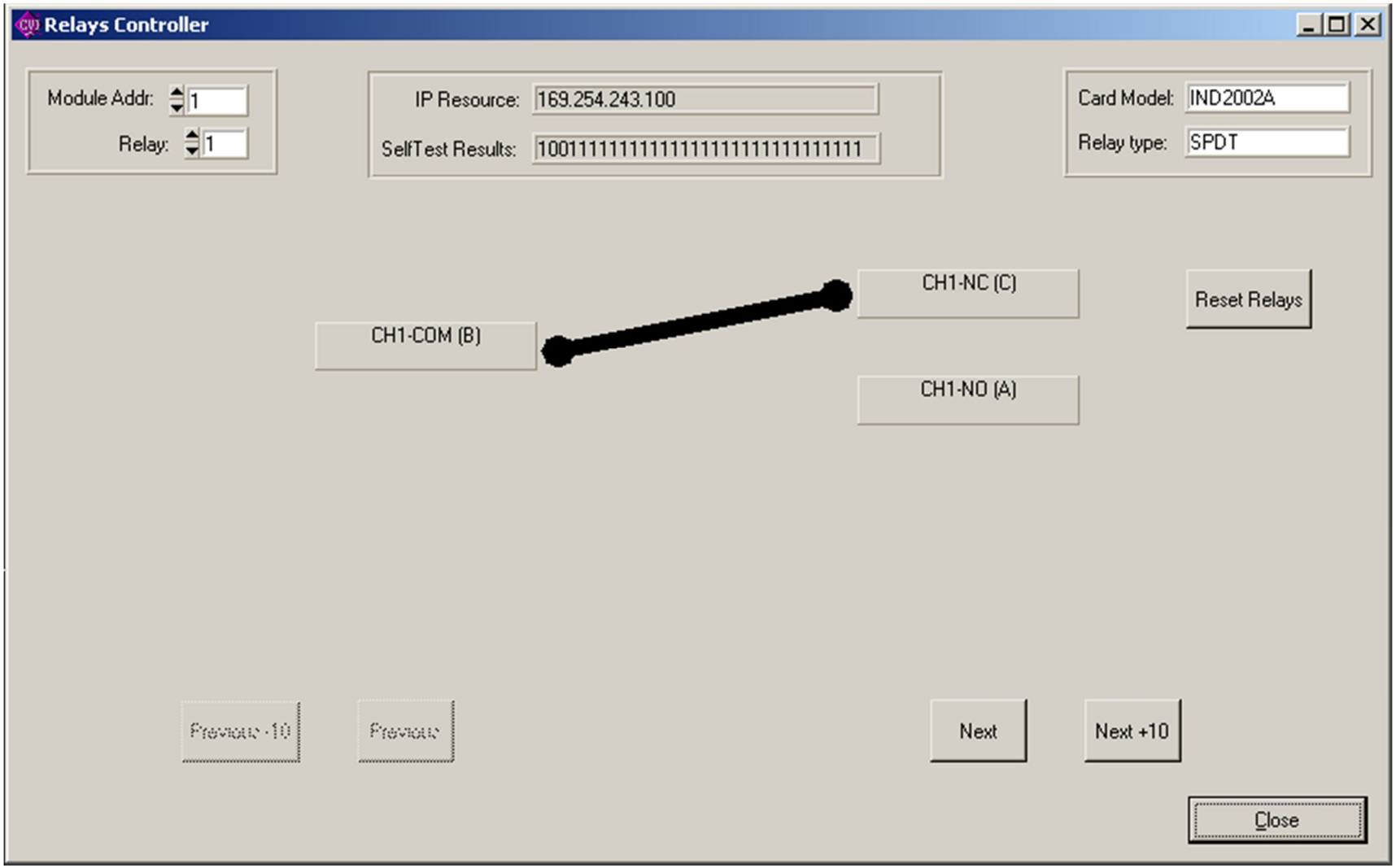


Figura 34. Panel principal del módulo de relés con dirección 01 (*IND2002A*)

En el panel principal se presentan únicamente los controles capaces de efectuar algún cambio (el resto se atenúan), así como sólo los valores válidos en los controles correspondientes.

Para controlar los relés de otro módulo es preciso interactuar con el control de dirección de módulo (*Module Addr*), bien mediante las flechas de las que dispone o en su lugar mediante la introducción de un nuevo dígito, en cualquiera de los dos casos solo es posible elegir un módulo operativo. El control del relé a manipular (*Relay*) funciona de forma similar, aunque también factible emplear los botones de control *Previous-10*, *Previous, Next* y/o *Next+10*.

En el supuesto de que se decida cambiar de posición, por ejemplo, el relé 26 del módulo con dirección 2, es necesario concretar primero el módulo y a continuación el relé. La *GUI* recoge la información del módulo y relés deseados (*funciones indrasmip_idn(..) y indrasmip_QueryRelay(…)*) y se muestra en el panel principal, como en la figura 35.

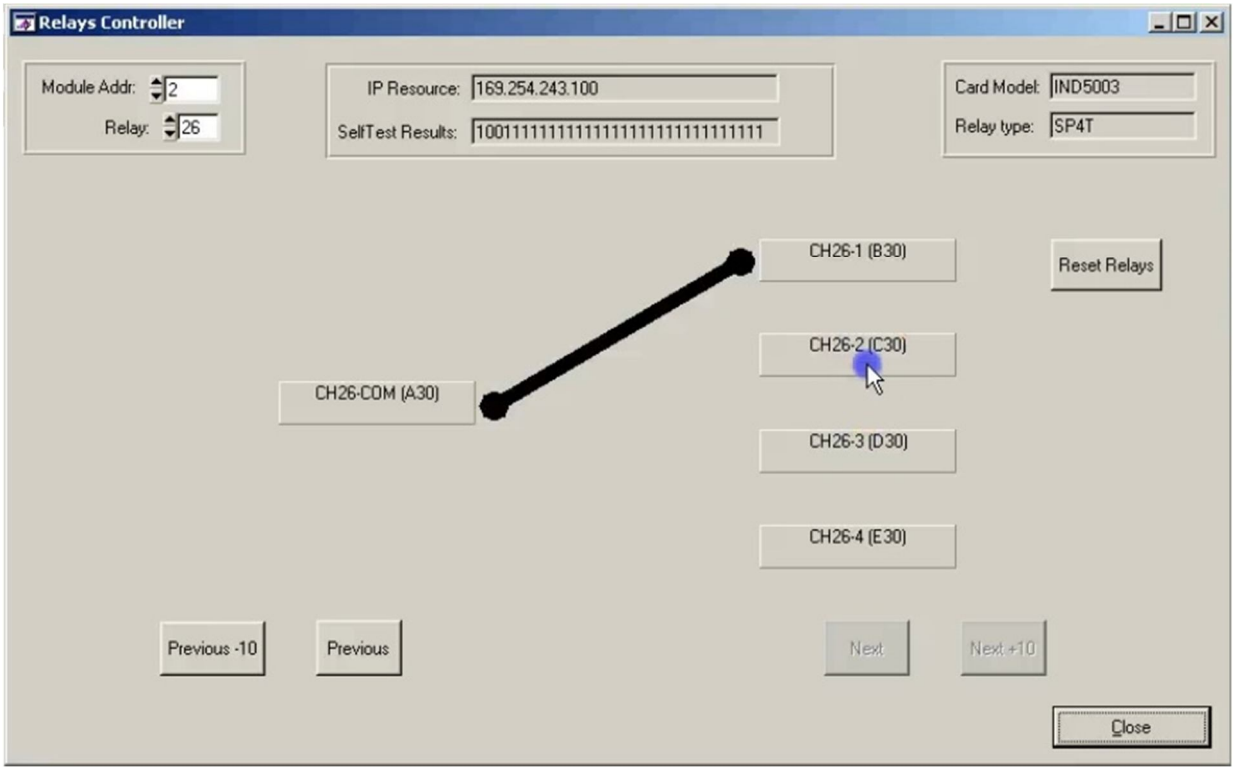


Figura 35. Panel Principal del módulo de relés con dirección 02 (*IND5003*) - Inicial

Se puede advertir que se pretende cambiar a la posición 1 (*CH26-2(C30)*), lo que resultaría en el uso de la función indrasmip_set_relay_list(…) del *driver*, que traduce el comando a *"M02R26P1\r\n"* para el *firmware*. El firmware procesa la orden y emite el *ACK*, interpretado como éxito de ejecución (*VI_SUCCESS*) hacia la *GUI*, que actualiza la visualización del relé, tal y como aclara la figura 36.

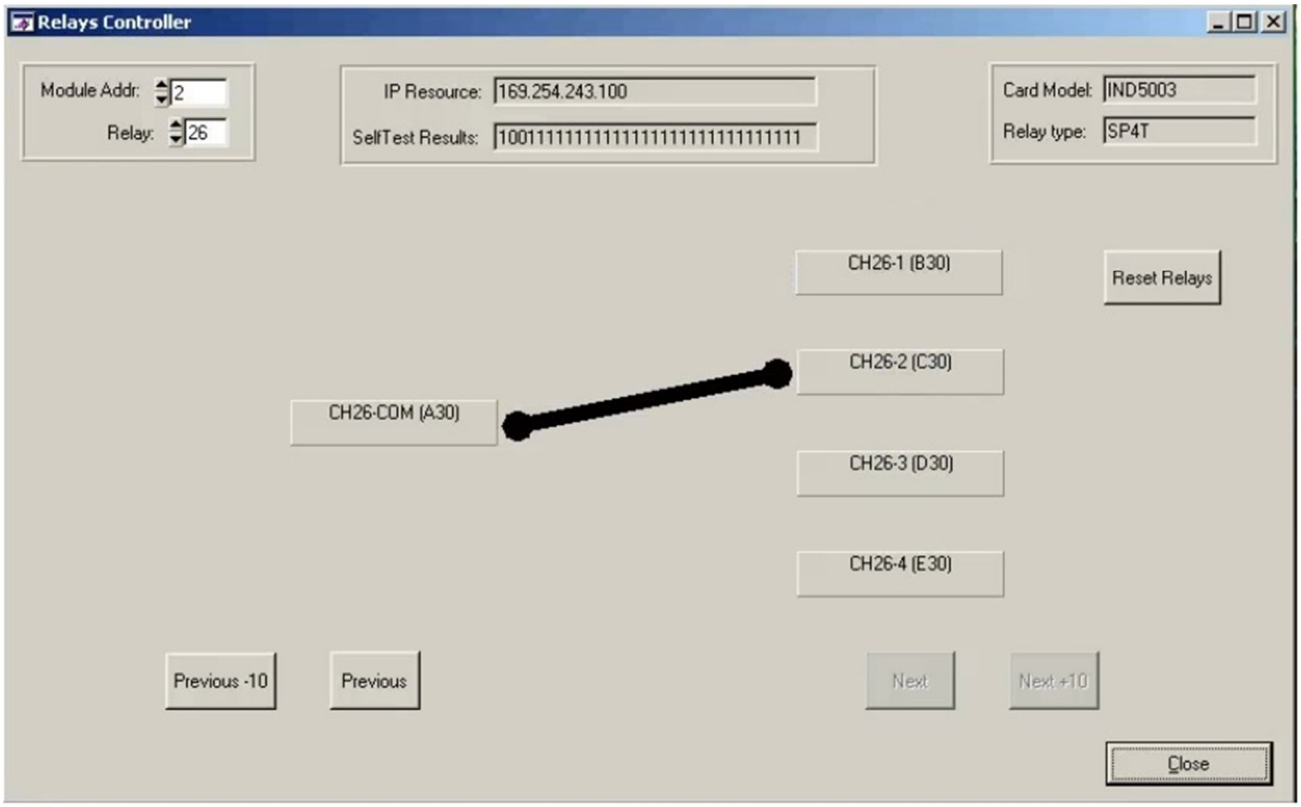


Figura 36. Panel Principal del módulo de relés con dirección 02 (*IND5003*) - Final

La GUI ofrece la posibilidad de resetear todos los relés de un módulo de relés de una vez, por medio del botón de control *Reset Relays*. Si se pulsa se realiza una llamada la función *indrasmip_reset(…)*, que envía el comando **RST* al módulo seleccionado y este retorna el *ACK*, que se transforma en *VI_SUCCESS*, entendible por la *GUI*.

El usuario puede finalizar en cualquier momento el control de relés, mediante el cliqueo sobre el botón *Close* del panel principal o sobre el icono X de la esquina superior derecha de la *GUI*, lo que lanza la función *indrasmip_close(..)* del driver, que a su vez efectúa un *ViClose(…)* sobre la interfaz *Ethernet-Serie,* que finaliza la sesión abierta (*ViOpen(…)*) concluyendo cualquier comunicación.

La figura 37 presenta el diagrama de secuencia de las funcionalidades de cerrar la interfaz (*Close*), conmutar un relé (concretamente la conmutación de R03 a P1 anteriormente detallada) y resetear todos los relés del módulo (*Reset Relays*).

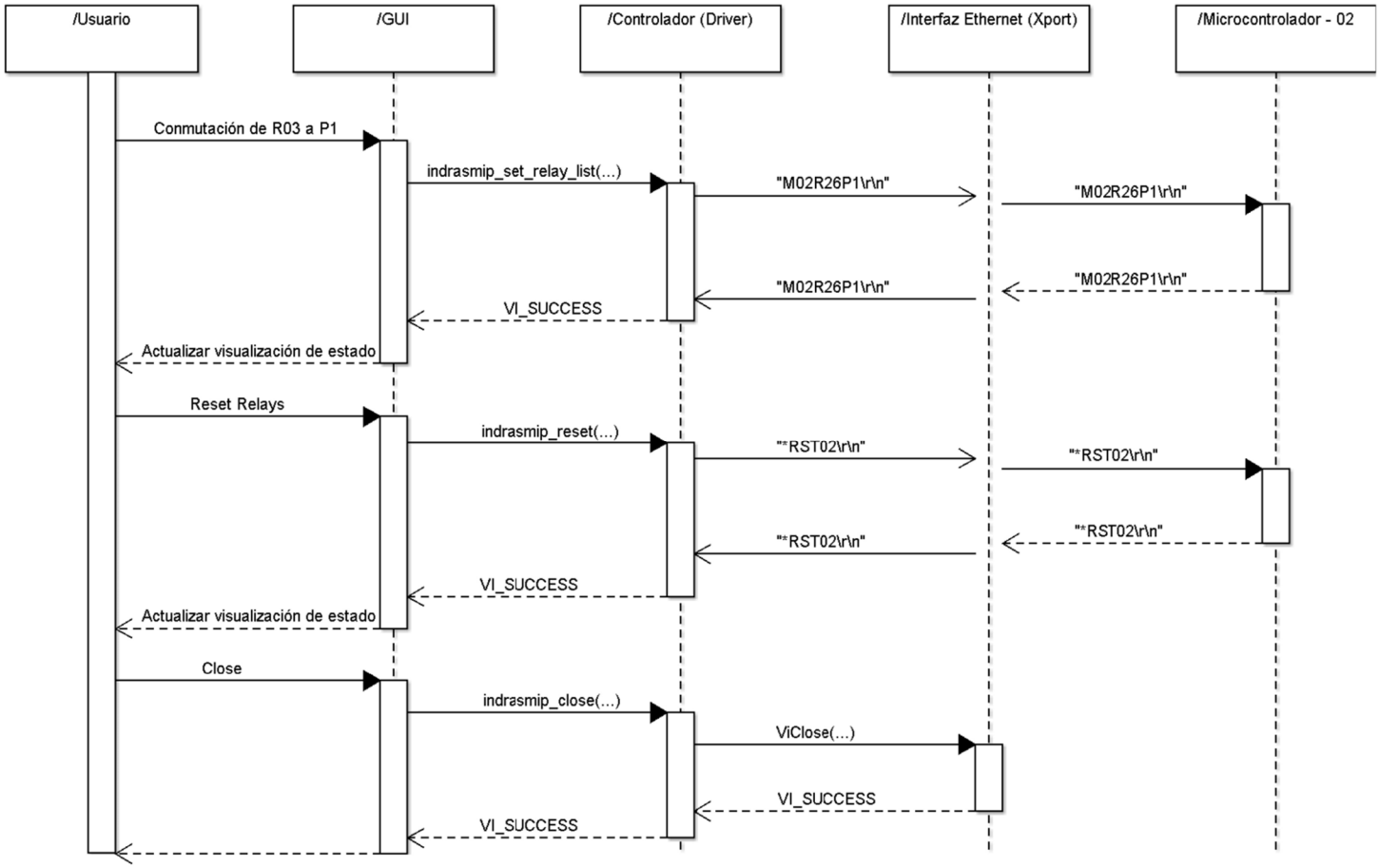


Figura 37. Diagrama de secuencia software – Funciones

Capítulo 5. Pruebas

Las pruebas del *hardware* individualmente resultan complicadas de desarrollar; sin embargo, el desarrollo del *software*, una vez implementado el *hardware*, por medio de una metodología *top-down* en colaboración con una constante depuración, goza de múltiples beneficios que acortan el tiempo de desarrollo y aumentan la calidad del sistema.

Así, se han efectuado numerosas pruebas que consistían en la verificación de pequeñas partes de la funcionalidad total, y tras comprobar cada una de estas fracciones se ejecutaron unas últimas pruebas globales que afectaban a todo el sistema.

5.1 Instalación y configuración del sistema

Aunque el diseño está orientado a su fabricación plenamente exterior, solamente las tarjetas de circuito impreso se han manufacturado en una empresa externa, mientras que los componentes *SMD* se han soldado manualmente en la compañía, lo que deriva en beneficios tales como la implantación y verificación del *hardware* por partes en los prototipos y el consecuente ahorro económico. Además para una mayor comodidad en las mediciones, no se han implantado conectores delanteros en un primer momento en ninguno de los módulos de relés como muestran las figuras 38, 39 y 40, lo que permite comprobaciones directamente sobre la *PCB* en lugar de en el conector.

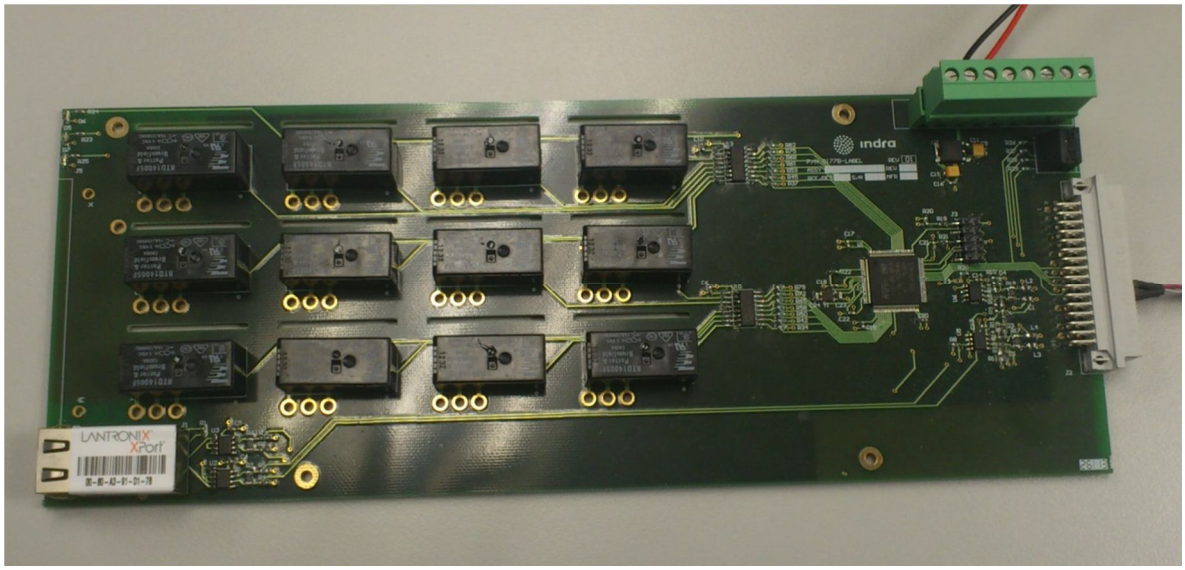


Figura 38. Módulo *IND2002A* - Frontal

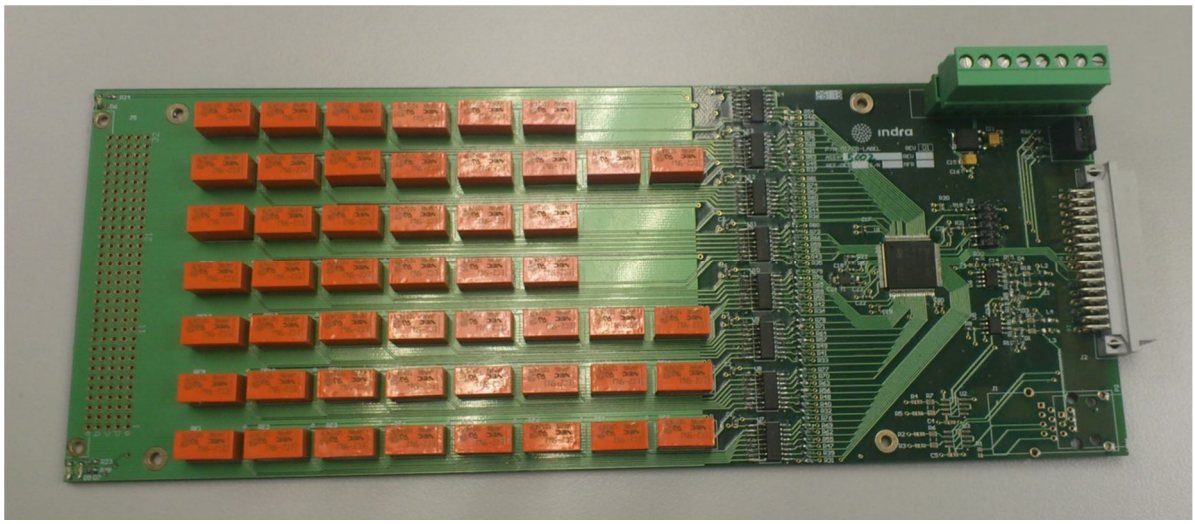


Figura 39. Módulo *IND5002* - Frontal

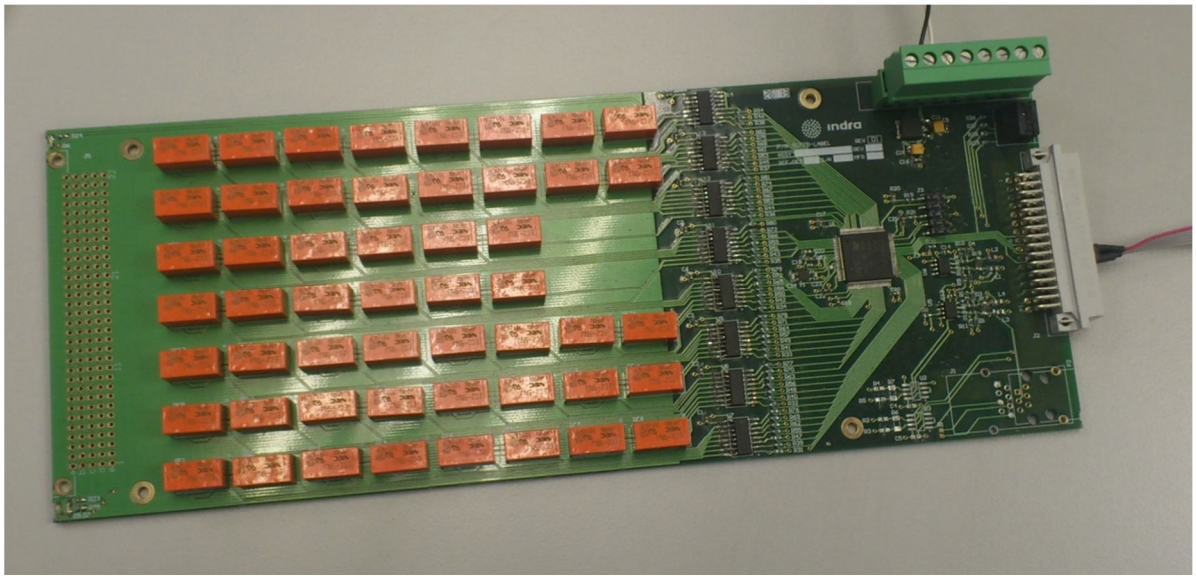


Figura 40. Módulo *IND5003* - Frontal

Si bien, la mayoría de componentes (*SMD*) se sueldan por la cara frontal de las tarjetas, existen componentes que emplean una fijación en la *PCB* más consistente, dado que representan conectores o elementos de mayor peso, y se implantan mediante su inserción, y a continuación su soldadura. Las figuras 41 y 42 ilustran la cara posterior de los módulos de relés (el módulo *IND5003* es prácticamente igual al *IND5002*) y dicho acabado.

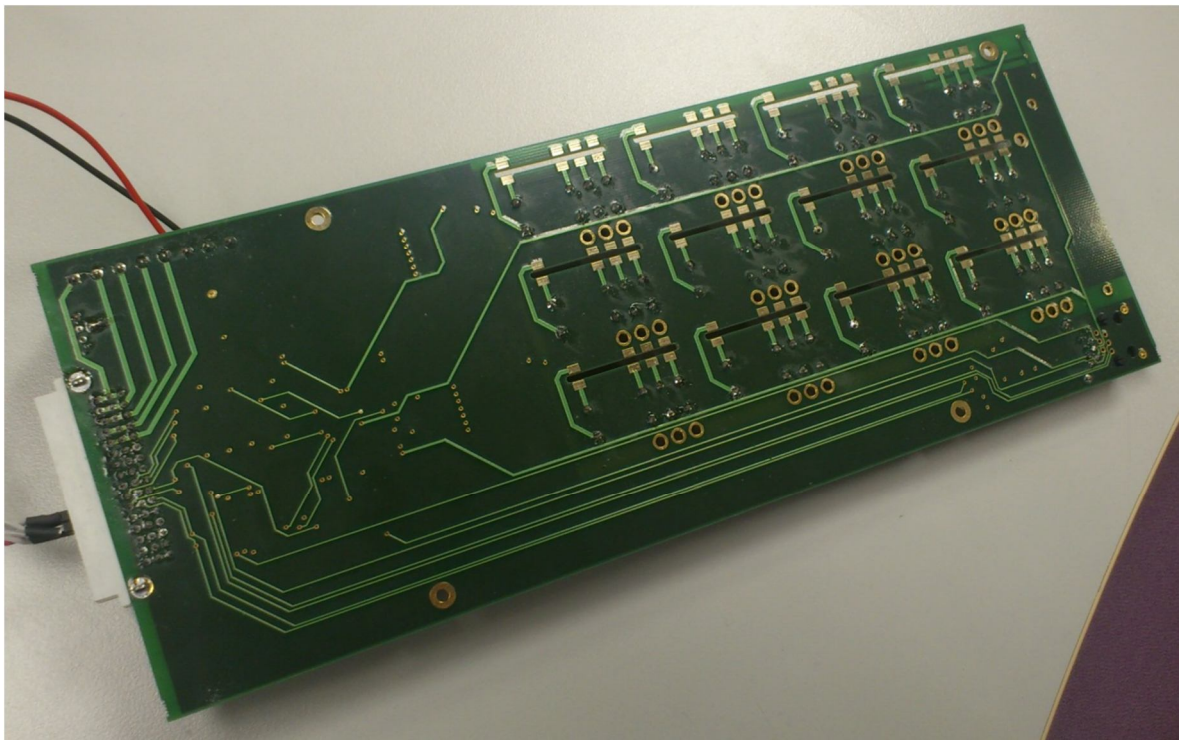


Figura 41. Módulo *IND2002A* - Posterior

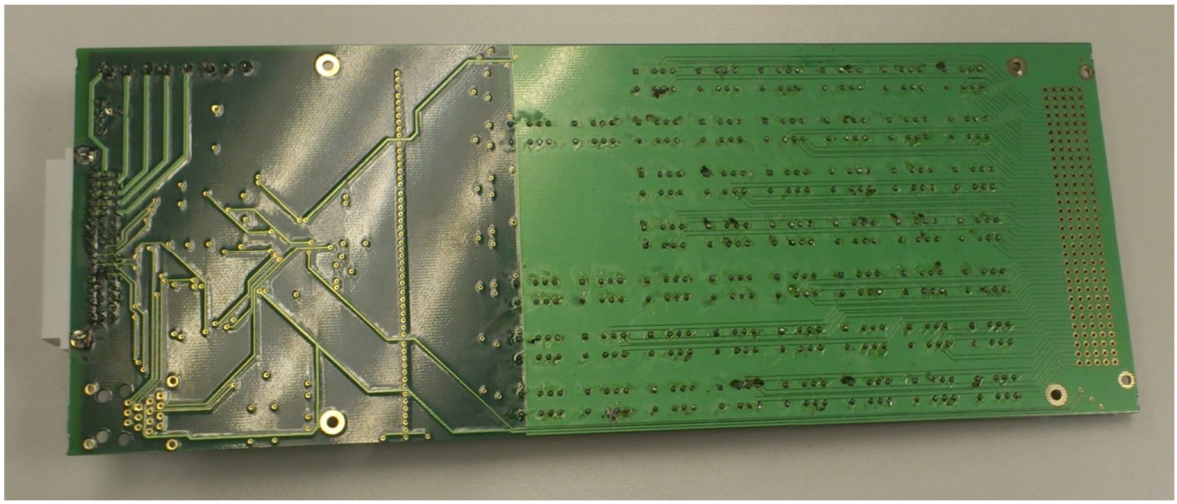


Figura 42. Módulo *IND5002* - Posterior

Se observa, tal y como se ha procedido en la fase de diseño del *hardware*, que las tarjetas presentan una gran parte común y diferenciaciones que convierten cada módulo en un tipo de tarjeta distinta. Los módulos *IND5002* e *IND5003* aparentemente son prácticamente iguales; sin embargo, la configuración funcional de los relés, además de que el módulo *IND5002* cuenta con dos relés menos, es la clave entre un diseño y otro. El módulo *IND2002A* puede resultar una tarjeta con pocos relés, pero soportan gran amperaje y tensión y de ahí su tamaño, por lo que se implementó una manera auxiliar de fijar los relés a la *PCB*, en caso de que en un futuro

se utilizasen relés de mayores dimensiones; a través de las ranuras es posible tumbar en horizontal los relés e insertar pequeñas *PCBs* soldadas a la base de los relés y unir las terminaciones de estas a las del módulo *IND2002A*.

Tras el establecimiento de todos los componentes necesarios, el sistema se encuentra preparado para los primeros ensayos, pero para llevar a cabo cualquier prueba, es necesario elaborar el conexionado del *PC* hasta el *microcontrolador*. Una parte de este cableado se encuentra determinado, el depurador *ST-Link Debugger,* y aunque otorga una conexión cableada *USB Mini-B a USB tipo A* para el ordenador es indispensable la conexión del depurador hasta el *microcontrolador*. [38]

El depurador ofrece una interfaz *JTAG/SWD*, y ya que *SWD* permite la grabación y depuración del *microcontrolador* a través de tan solo dos señales (*PROG_ SWCLK* y *PROG_SWIO*), junto con la alimentación *(+3.3VDC* y *GND*), se implementa el cableado de 4 señales entre ambos dispositivos.

En la elaboración del cable hay que advertir el conector de la *PCB (Molex 15-91-2100)* y el conector *SWIM/JTAG/SWD*. El conector complementario de la *PCB* (*Amphenol 84281210030134*) admite un cable plano de 10 conductores (*3M 3365/10*), pero el depurador incluye un conector *IDC* de 20 pines con su correspondiente cable plano y cabeceras, por lo que únicamente es imprescindible reestructurar las 4 señales necesarias mediante un empalme entre los dos cables planos. Es conveniente el uso de un conector macho en la cabecera del cable plano del depurador, ya que de otro modo es complicado realizar un empalme sobre la cabecera hembra; es por ello que se utiliza el alojamiento de cabecera *3M 961220-6404-AR* adecuado para esta cabecera en concreto, y sobre este conector se realiza el empalme de manera más cómoda.

La tabla 43 presenta la información pertinente acerca de las señales de los pines del conector del depurador *ST-Link Debugger*.

Pin N°	Name	ST-LINK function	Target connection
1	TVCC	Target VCC	MCU VCC
2			
3	TRST	GROUND	GND
4	UART-RX	Unused	GND or not connected
5	TDI	JTAG TDO, SWO	TDI
6	UART-TX	Unused	GND or not connected
7	TMS	JTAG TMS, SW IO	TMS, SWIO
8	BOOT0	Unused	GND or not connected
9	TCK	JTAG TCK, SW CLK	TCK, SWCLK
10	SWIM	Unused	GND or not connected
11	NC	Not connected	Not connected
12	GND	GROUND	GND
13	TDO	JTAG TDI	TDO
14	SWIM-RST	Unused	GND
15	RESET	RESET	RESET (optional)
16	KEY	No pin	Not connected
17	NC	Not connected	Not connected
18	GND	GROUND	GND
19	VDD	VDD (3.3V)	Not connected
20	GND	GROUND	GND

Tabla 4. Conexiones del cable *JTAG* [38]

El conector de la *PCB* (*Molex 15-91-2100*) dispone de las señales *+3.3VDC*, *GND, PROG_SWIO* y *PROG_SWCLK* ubicadas en los pines 2, 6, 8, y 4 respectivamente (ver plano *IND-6* del anexo A.5.2), que enlazan con las señales del cable *JTAG* correspondientes.

En la tabla 43 se observa que la alimentación de la *PCB* debe estar conectada a los terminales 1 y 2 (*TVCC*), pero el pin 19 (*VDD*) es capaz de proporcionar alimentación, por lo que sería conveniente realizar una conexión divisible entre los pines 19 y 1 y 2, de forma que al carecer la *PCB* de la alimentación, sería posible alimentar por medio del depurador el *microcontrolador* y realizar pruebas, depuraciones o grabaciones.

Así, se establece una unión entre los cables, tal y como se muestra en la figura 43:

- Pin 2 del conector de la *PCB* (*+3.3VDC*) al terminal 1 del conector del depurador (*TVCC*).
- Pin 19 del conector del depurador (*VDD*) al terminal 1 del mismo conector (*TVCC*), de modo que pueda separarse.
- Pin 4 del conector de la *PCB* (*PROG_SWCLK*) al terminal 9 del conector del depurador (*TCK*).
- Pin 6 del conector de la *PCB* (*GND*) al terminal 12 del conector del depurador (*GND*).

- Pin 8 del conector de la *PCB* (*PROG_SWIO*) al terminal 7 del conector del depurador (*TMS*).

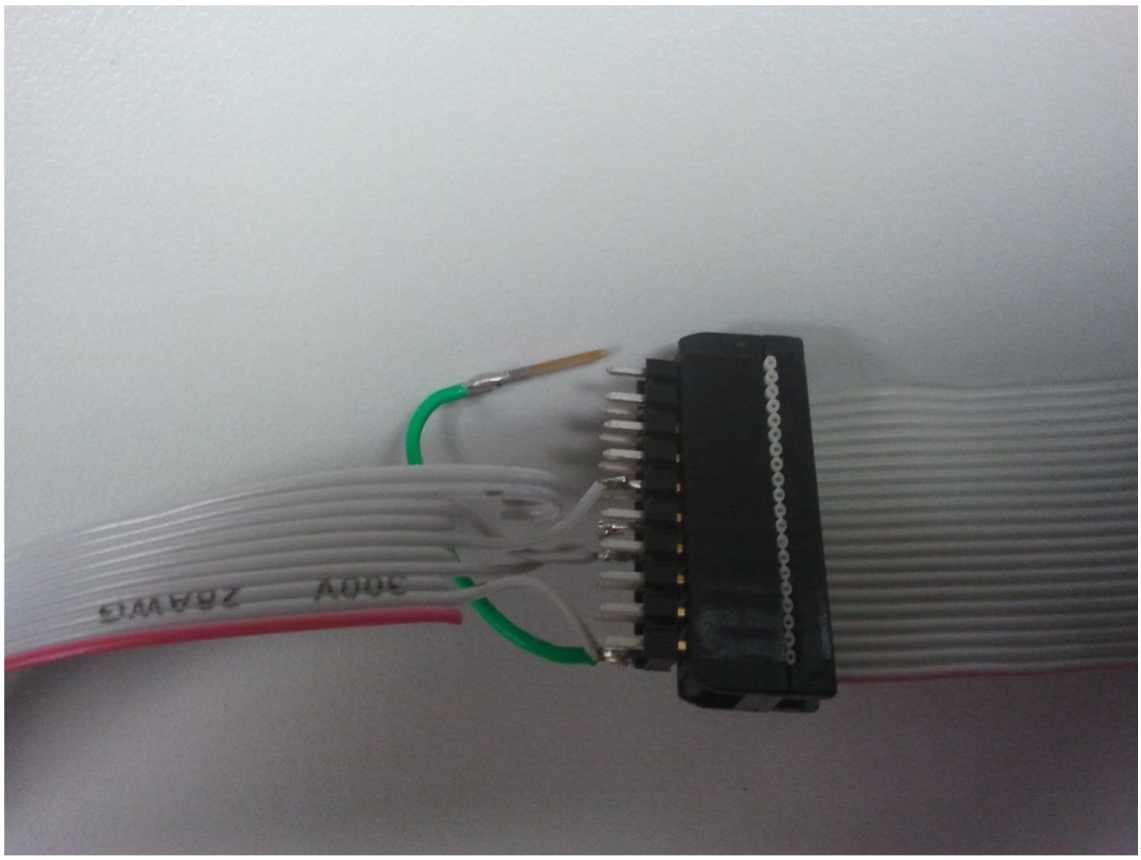


Figura 43. Empalme de los cables planos

Aunque el cable implementado se encuentra lejos de la ostentosidad, logra los objetivos para los que ha sido diseñado, por lo que resulta una buena alternativa para el desarrollo del sistema.

Cabe la posibilidad de que el sistema operativo no posea el *driver* necesario para detectar y manejar el depurador *ST-Link Debugger*, por lo que se requiere la instalación de una versión reciente del mismo suministrada en un *CD* junto al depurador. [38]

De esta manera, el cableado del *PC* hasta la *PCB* se halla completamente implementado y listo para la conexión, como se aprecia en la figura 44.

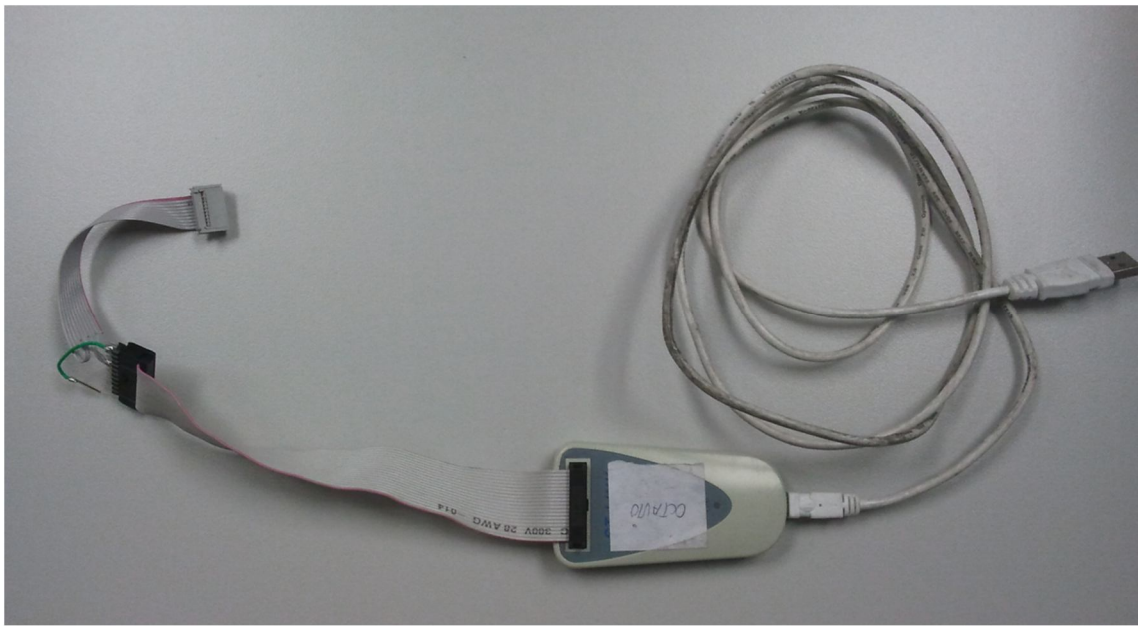


Figura 44. Cableado *PC – Microcontrolador*

La depuración y grabación del *microcontrolador* requiere de la conexión de la *PCB* al *PC*, y tal y como se encuentra configurado el entorno de desarrollo *µVision*, solo es necesario bien pulsar *Start/Stop Debug Sesión* en el menú *Debug* para comenzar la depuración, o en su lugar presionar el botón *LOAD* para la grabación del *firmware* en la memoria *Flash* del *microcontorlador*. La depuración constante del *firmware*, a través de estos medios, facilita la labor del desarrollador, ya que aísla los fallos significativamente más rápido y en caso de una funcionalidad impropia la solución resulta de examinar la última actualización, reduce el tiempo de desarrollo y garantiza la calidad del *firmware* a través de múltiples depuraciones.

El desarrollo del *driver* y la *GUI* también se ha efectuado directamente sobre el *hardware*. La prueba y depuración de estas capas de software requiere de la coexistencia parcial de la parte funcional correspondiente de ambos, así como de una configuración previa de la interfaz *Ethernet-Serie* (*XPort*), la instalación de un paquete de software con las librerías y utilidades necesarias para emplear el estándar *VISA* en el sistema (*NI_VISA*) [19], y la implantación de un sistema de alimentación a 5V (una fuente de alimentación regulable).

Para llevar a cabo la configuración de *Xport* es imprescindible en primer lugar detectar el dispositivo en el sistema operativo conectado a través de un cable *Ethernet* cruzado, lo que se resuelve mediante la instalación del software proporcionado con la interfaz *Ethernet –Serie* (*DeviceInstaller*). Dado que *Xport* presentará una dirección *IP* fuera de rango, problema que se resuelve mediante la modificación de la dirección *IP* del dispositivo, hay que hacer uso de *DeviceInstaller* de *Xport*. Se inicia el programa y se presiona sobre el icono de asignar *IP* (*Assign IP*), a continuación si se requiere se introduce la dirección física del *Xport*, después se asigna un dirección *IP* específica (*Assign a specific IP address*) sin ocupar dentro del rango del ordenador (los tres primeros campos de la dirección IP deben ser iguales) y por último se valida la configuración (*Assign* y *Finish*). [15]

Xport incorpora *Web Manager*¸ es por ello que permite la configuración a través del explorador *Web*. En el explorador de internet del sistema operativo se introduce la dirección *IP* del dispositivo, establecida con anterioridad, lo que deriva en el acceso a la configuración de la funcionalidad de *Xport*, donde la única modificación realmente interesante en este sistema es la conformación de los factores que rigen la comunicación serie *RS-232* (pestaña *Serial Settings*).

Xport realiza una transformación de protocolos en la información (*Ethernet-Serie*), pero dada la diversidad de formatos dentro de los estándares serie *RS-232* (paridad, bits de parada, velocidad, etc.) es preciso compartir los mismos parámetros del *firmware* en cuanto a la comunicación serie para lograr un intercambio de información efectivo:

- Velocidad de 115200 baudios.
- Sin Paridad.
- Sin Control de flujo.
- 8 bits de datos
- 1 bit de parada.

La figura 45 ilustra el administrador *Web* de la configuración del *Xport* y los campos modificados en *Port Settings* de acuerdo a las variables que condicionan la comunicación serie *RS-232*.

El estándar *Ethernet* ofrece una comunicación más universalizada y auto-detectable (ver Anexo A.2.2), por lo que no es necesario estudiar ninguna configuración al respecto.

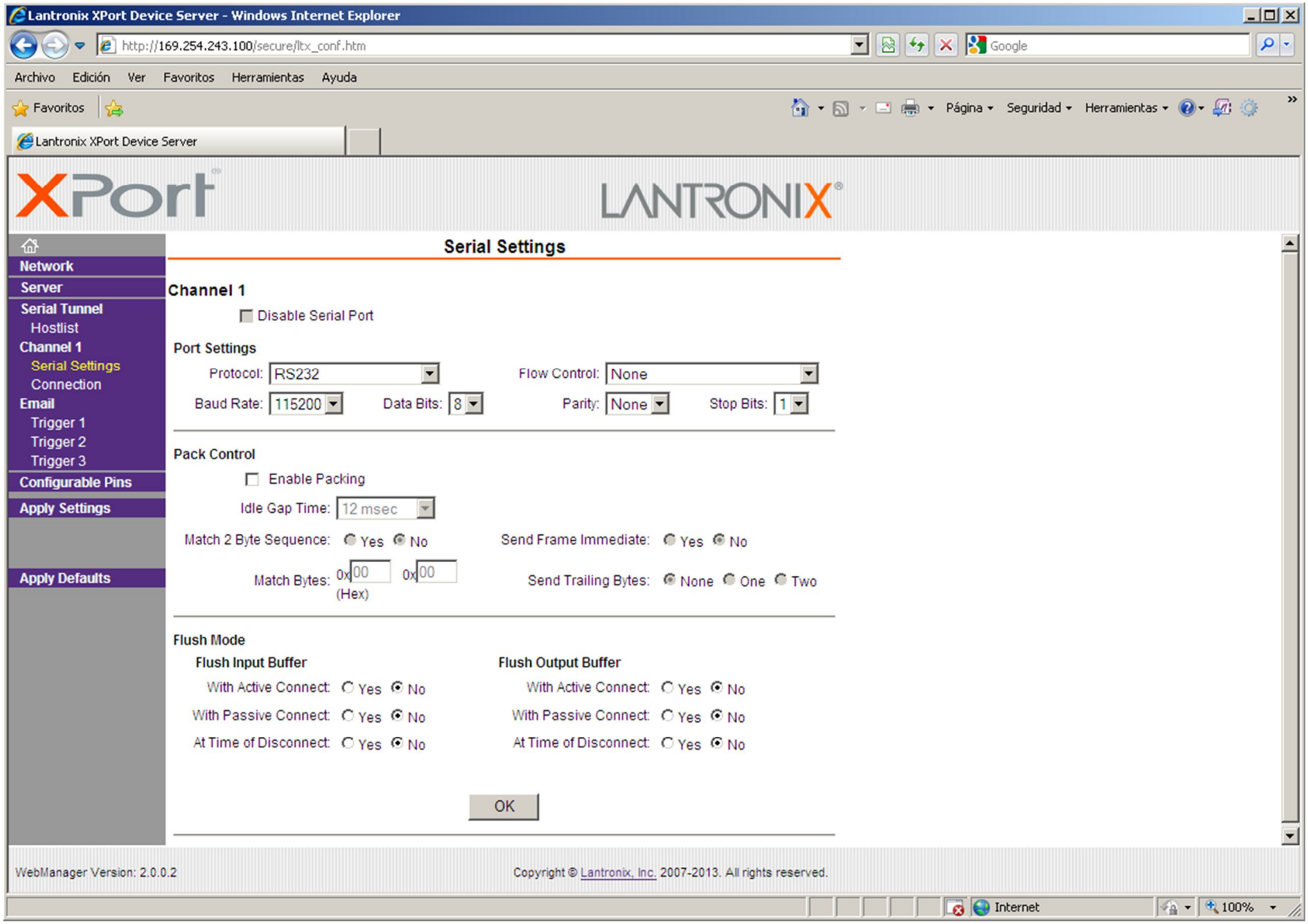


Figura 45. Configuración del *Xport* en *Web Server*

La implementación del estándar *VISA I/O* de *National Instruments* (*NI-VISA*) incorpora todas las librerías de software, utilidades interactivas (*NI I/O Trace* y *VISA Interactive Control*) y programas de configuración a través de *Measurement & Automation Explorer*, lo que resulta no solo imprescindible para las depuraciones y pruebas del driver y la GUI, sino que constituyen una gran ayuda en el desarrollo. [20]

NI I/O Trace es una aplicación que monitoriza, graba y reproduce las llamadas de las *API* de *National Instruments*, es por ello que se ha usado para localizar y analizar rápidamente cualquier llamada errónea por parte de las capas del *driver* y la *GUI*, así como para verificar una comunicación correcta con los instrumentos.

VISA Interactive Control es una utilidad que proporciona acceso a toda la funcionalidad de *VISA* interactivamente, encuentra automáticamente todos los recursos del sistema al iniciar (*VISAIC*) mediante el *Resource Manager* e intenta recopilar información de ellos. Permite iniciar sesiones de comunicación con los instrumentos, por lo que resulta una buena herramienta de validación y prueba de los comandos del *firmware* con *VISA*. Aunque no se ha empleado apenas, constituye un conveniente punto de partida para iniciarse en el desarrollo con *VISA*.

Measurement & Automation Explorer es una interfaz gráfica de usuario que suministra acceso a todos los instrumentos detectables para configurarlos, ejecutar diagnósticos y paneles frontales, visionar los dispositivos e instrumentos del sistema, actualizar el software de *National Instruments*, crear y editar canales, tareas, interfaces, etc. No resulta de gran utilidad en el desarrollo, pero conforma una herramienta muy provechosa en sistemas ya implementados con múltiples instrumentos conectados.

El sistema de alimentación implementado en el sistema provee al sistema de una alimentación estable de 3.3 V; sin embargo, algunos elementos junto el propio sistema de alimentación operan a 5V, por lo que es necesario emplear algún tipo de suministro de energía de este voltaje. En las pruebas y depuraciones se ha utilizado una fuente de alimentación *Adler XPD 33-16* (33V y 16A), más que suficiente para su propósito, que abastece el sistema por medio de unos cables conectados a los conectores complementarios (*Phoenix 1792304*) a los conectores de alimentación de la *PCB* (*Phoenix 1755794*), como se observa en la figura 46. [29]

En las primeras pruebas y depuraciones no es necesario emplear todos los elementos del sistema, es por ello que resulta de gran ayuda trabajar sobre un único módulo de relés que albergue el *Xport*, y puesto que en el resto de pruebas la interfaz *Ethernet-Serie* se implanta en el módulo *IND2002A*, es adecuado ensayar sobre el módulo distinto con mayor número de relés (*IND5003*) con el correspondiente *Xport*, pero una vez implementadas todas las facetas del sistema beneficia un conjunto de pruebas sobre varios módulos de relés.

Dado que el proyecto no se dedica al estudio de un *mainframe* o *backplane* que conecte los módulos de relés entre sí, es imprescindible elaborar algún medio que permita la conexión de diversos módulos con una única interfaz *Ethernet-Serie*, de manera que simule un entorno con varios módulos conectados y las pruebas gocen de mayor realismo. En un principio se ha utilizado un pequeño cable plano que conecta en paralelo las señales *ETH_RS485_TX_B, ETH_RS485_TX_A, ETH_RS485_RX_A* y *ETH_RS485_RX_B* (pines *C10, A11, B11* y *C11* respectivamente) del conector *J2* trasero (*DIN-048CPC-SR1*), como muestra la figura 46 entre

los módulos *IND2002A* e *IND5003*, y más tarde se ha empleado un empalme en el mismo cable plano para conectar el tercer tipo de módulo de relé.

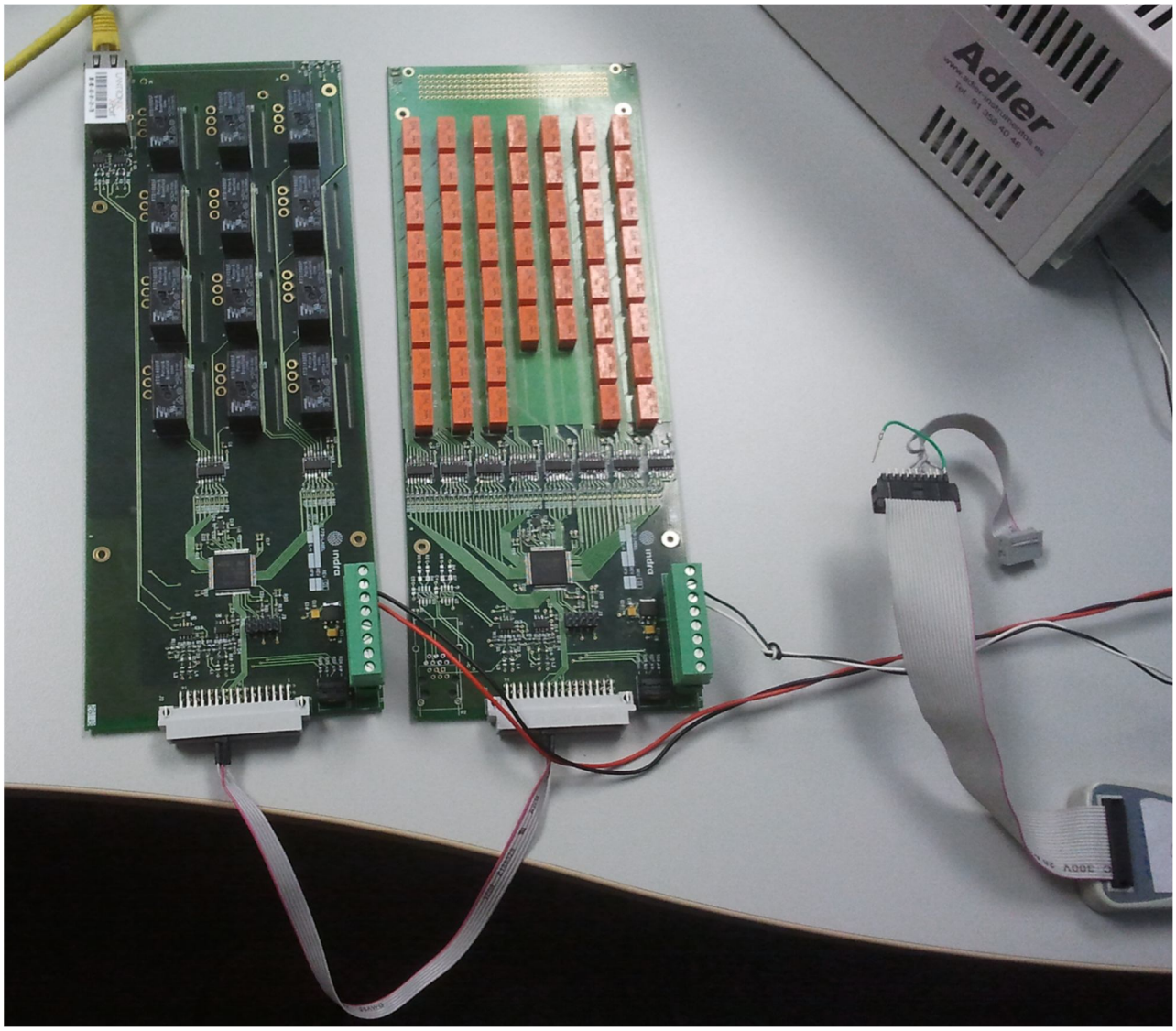


Figura 46. Conexionado y cableado para de los módulos de relés *IND2002A* e *IND5003*

5.2 Descripción de las pruebas

Las pruebas constituyen una fase necesaria en el proceso de desarrollo del sistema, dado que permiten identificar la naturaleza de los fallos y actuar en consecuencia para eliminarlos; sin embargo las pruebas finales conforman un medio de estudio de la consistencia e integridad del sistema, además de demostrar la veracidad de los fundamentos en los que se ha basado el diseño. Para llevar a cabo las pruebas correspondiente es conveniente simular el entorno en el que se encontrará el sistema en las condiciones más realistas alcanzables, por lo que se

conectan los distintos tipos de módulos de relés entre sí a una única interfaz gráfica *Ethernet-Serie,* que a su vez es conexionada mediante un cable cruzado *Ethernet* al ordenador, donde se manejan los relés a través de la interfaz gráfica.

Cabe la posibilidad de que durante el diseño del hardware se haya cometido algún error en el rutado de pistas en cada módulo de relés, es por ello que conviene asegurar la implementación del *hardware* a través de un estudio de todas las posiciones de los relés conmutando cada una de ellas y examinando los cambios en los conectores delanteros *GMCT41M* y *DIN 41612 004778*, por medio de un instrumento adecuado como un multímetro en su función de análisis de continuidad (pitido). Esta prueba determina una correcta funcionalidad global del módulo de relés, por lo que se deduce un ensayo indispensable.

Uno de los requisitos establecidos consiste en garantizar la ausencia de señales en posiciones de relé indeseadas, especialmente en los módulos de relés con configuraciones *SPQT*, ya que su implementación por medio de tres relés funcionales en dos relés físicos puede provocar la aparición de tensiones o corrientes. A través del *firmware* se soluciona la posible existencia de estas señales, pero es preciso efectuar una comprobación de seguridad. La prueba se lleva a cabo conectando en cada una de las cuatro posiciones del relé de un módulo *IND5003* una sonda de osciloscopio, aplicando una señal continua de 5V en el terminal común y conmutando el relé para examinar cualquier cambio de tensión en las posiciones del relé.

Por último, se realiza una batería de pruebas sobre los relés, que implique su conmutación repetidamente (20 veces por relé y posición) a fin de garantizar la fiabilidad del sistema.

5.3 Resultados

En un primer momento los resultados ofrecidos sobre las pruebas que intervenían en el desarrollo del sistema otorgaban fallos y errores que permitían la evolución del sistema, pero concluida esta fase, el resultado de las pruebas finales has demostrado que el diseño, tanto *hardware* como *software* se ha implementado correctamente.

El estudio de cada uno de los pines de los conectores delanteros es acorde a los requisitos técnicos establecidos y la conmutación de cada uno de los relés a dichas posiciones se ejecuta con éxito, por lo que la funcionalidad del sistema es completa y precisa.

El análisis de señales en posiciones indebidas revela que la implementación del *firmware* basta para evitar este problema. En la figura 47 se han conectado a los cuatro canales de un osciloscopio cada una de las posiciones del relé, de manera que su número de posición coincide con el canal. En un principio la señal se encuentra aplicada en la posición 1 y se conmuta a la posición 4.

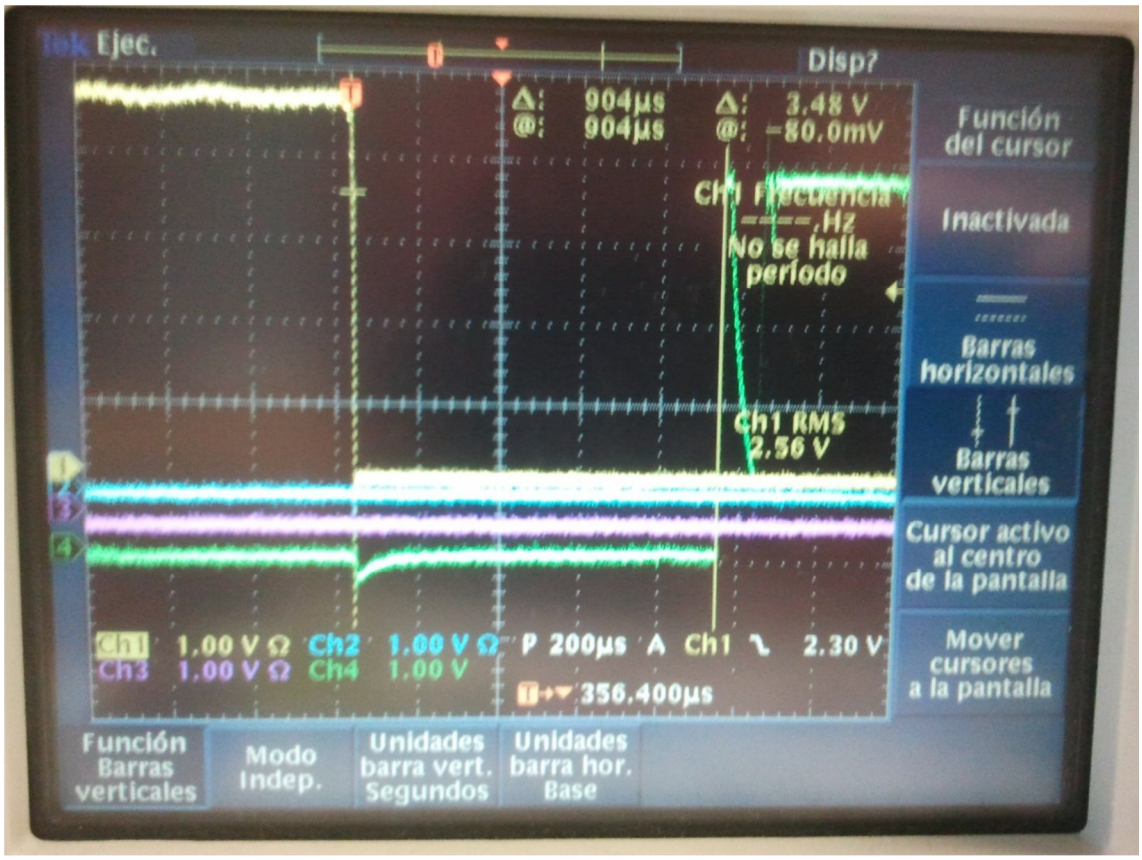


Figura 47. Prueba de señales indeseadas conmutando de la posición 1 a la posición 4

En la figura anterior se aprecia como en la posición 1 se halla aplicada una señal continua de casi 5V y al efectuar la conmutación hacia la posición 4, transcurren 904µs (valor acorde al conformado en el *datasheet* del relé [26]) hasta que vuelve a aparecer al señal en el canal 4, y durante este lapso de tiempo el resto de canales permanecen inalterables.

Si bien puede parecer un caso aislado se ha repetido la prueba numerosas veces ejecutando distintas conmutaciones obteniendo resultados similares, como en la figura 48. La señal está desde un principio en el canal 4 y se conmuta hacía la posición 1, lo que demora 708µs.

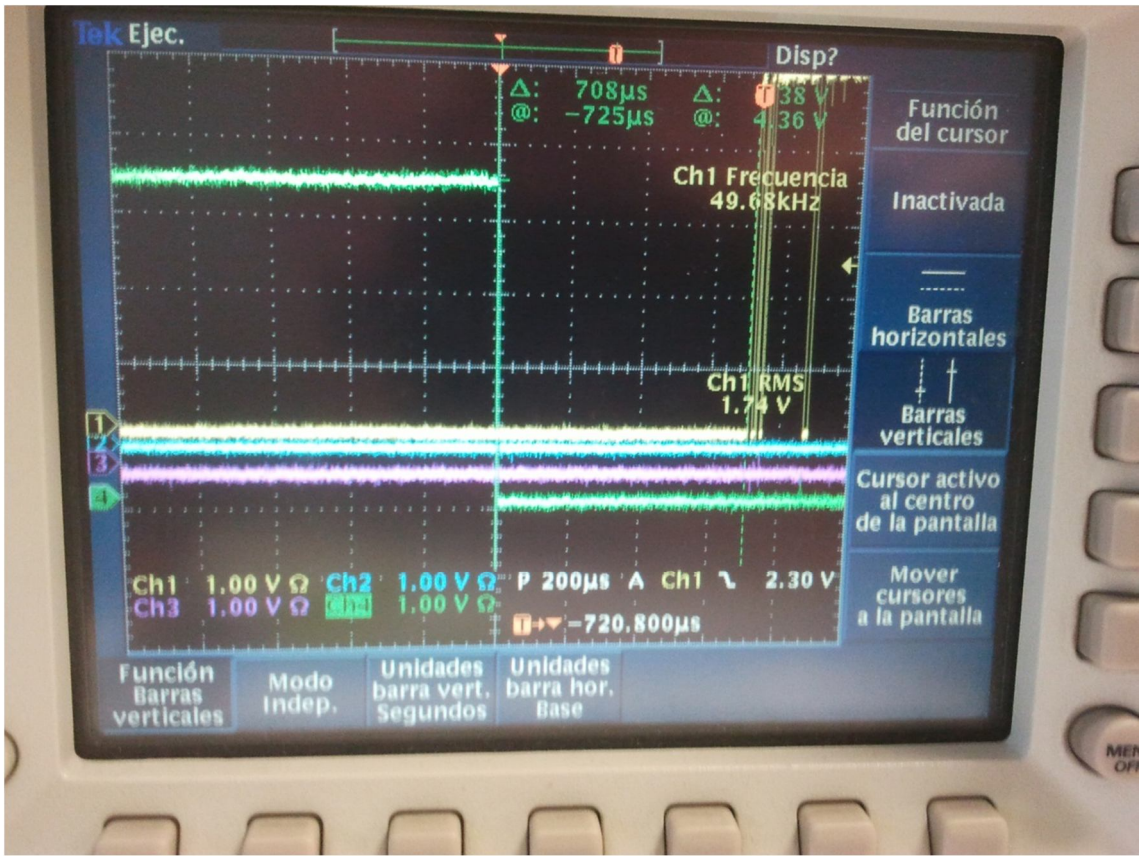


Figura 48. Prueba de señales indeseadas conmutando de la posición 4 a la posición 1

En ambas figuras (47 y 48) se aprecian cambios bruscos en el voltaje de la posición final conmutada, esto es debido a los rebotes que efectúa la varilla al entrar en contacto con el terminal, algo habitual en los relés, por lo que esta prueba constituye un ensayo fehaciente de la funcionalidad y la formalización de requisitos.

Las baterías de pruebas sobre los relés, que radican en la reiteración de conmutaciones sobre las posiciones los relés, hasta 20 veces por cada posición, derivan en la conclusión de un sistema fiable y eficaz.

Capítulo 6. Presupuesto

El presupuesto hace referencia al desarrollo y construcción del sistema que se ha llevado a cabo y tiene en cuenta los recursos *hardware*, *software* y humanos. Aunque se encuentra estrechamente ligado al proyecto, permite valorar el coste que supone el sistema frente a otros instrumentos de conmutación de relés que se encuentran actualmente en el mercado (precio en torno a 1500€ el módulo, o 9000€ un instrumento completo), y por tanto constituye una herramienta para un análisis de viabilidad económica.

6.1 Coste de los recursos *hardware*

El coste de los recursos *hardware* engloba todo el material dedicado a la implementación de los módulos de relés, así como a estos mismos. Dado que únicamente se han empleado tres prototipos de módulos de relés, un depurador *ST-Link Debugger* y el correspondiente cableado, elementos como la fuente de alimentación y el ordenador se obvian por su universal disponibilidad y por resultar innecesarios tras el desarrollo.

El coste de los módulos de relés es fraccionado en tres campos, uno por cada tipo de tarjeta de circuito impreso. Únicamente se incluye el precio de un bloque Conversor *Ethernet-RS485* en el módulo de relés *IND2002A*, pues constituye el módulo en el que se implanta preferentemente, hay que tener en cuenta que en un sistema con numerosos módulos (hasta 32) solo se emplea un único bloque, por lo que el coste asociado al conjunto *hardware* es ligeramente menor. El conector *P2 FCI 74543-3661LF* no se ha implantado, ya que se inserta prioritariamente en los módulos *IND2002A* y solo se ha ensayado con un único módulo *IND2002A* ya provisto de interfaz *Ethernet-Serie*, pero constituye un buen medio para la conexión de tarjetas de conjuntos de módulos sin el sobrecoste del *Xport* a un precio muy asequible (1,19 €).

Cada uno de los presupuestos individuales de los tipos de módulos de relés se encuentra desglosado en los componentes que los conforman, de acuerdo a su precio aproximado por los distribuidores de electrónica y las empresas de fabricación de tarjetas de circuito impreso.

Existen algunos componentes, como *R17* y *R18*, que solo se establecen en un único módulo (*IND5003* en este caso) del sistema, ya que se trata de las resistencias terminadoras del *RS-485*, es por ello que aunque apenas varían el coste conforman elementos a ser considerados particularmente en la implantación. *J4* tampoco se ha utilizado, ya que es un abierto del tamaño de una resistencia que se coloca en caso de que la cantidad de módulos de relés sobrepase los 16.

Las tablas 5, 6 y 7 muestran el coste detallado de todos los componentes de cada uno de los módulos de relés.

IND2002A						
Part Number del fabricante	Cantidad	Ud medida	Designador	Descripción	Precio/ud	Precio total
VJ0603V104ZXXCW1BC	20	ud	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C12, C13, C14, C16, C17, C18, C19, C20, C21, C22	Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 0.1uF 25volts Y5V +80-20%	0,050 €	1,000 €
TAJB106K010RNJ	2	ud	C11, C15	Capacitadores de tantalio - SMD sólido 10volts 10uF 10%	0,107 €	0,214 €
VJ0603A120FXACW1BC	2	ud	C23, C24	Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 12pF 50volts C0G 1%	0,050 €	0,100 €
VESD05A1A-HD1-GS08	4	ud	D1, D2, D3, D4	Supresores de ESD 5.0 Volt 16 Amp	0,406 €	1,624 €
APA2106SURCK	1	ud	D5	LED estándar - SMD Hyper Red R.A. 635nm, 250mcd	0,085 €	0,085 €
APA2106SYCK	1	ud	D6	LED estándar - SMD Yellow 590nm Water Clear 150mcd	0,093 €	0,093 €
APA2106CGCK	1	ud	D7	LED estándar - SMD Green 570nm Water Clear 60mcd	0,101 €	0,101 €
XP1001000-05R	1	ud	J1	Módulos de Ethernet XPort XE Ext. Temp. without Encryption	43,930 €	43,930 €
86093487313H55ELF	1	ud	J2	Conectores DIN 41612 48P R/A PLUG DIN 41612	1,300 €	1,300 €
MC114N/AA	37	ud	-	Pines de tamaño 16 para uso con AWG16 y GMCT41M, compatible con RoHS	0,660 €	24,420 €
15-91-2100	1	ud	J3	Alojamientos de cables y cabecera C-GRID 10 CKT BKWY H	1,050 €	1,050 €
GMCT41M0T0000	1	ud	J5	Conector de 41 contactos, macho, conexión multi-forma de panel y bastidor, crimpado, press fit, soldadura y aislado	14,950 €	14,950 €
MPZ2012S101A	4	ud	L1, L2, L3, L4	Gránulos, chips y matrices de filtros EMI Ferrite Chip Beads	0,058 €	0,232 €
1755794	1	ud	P1	Bloques terminales conectables 5.08 8P HEADER 180DG	1,670 €	1,670 €
RK73B1JTTD472J	8	ud	R2, R3, R4, R5, R9, R10, R11, R12	Resistores de película gruesa - SMD 1/10watt 4.7Kohms 5%	0,050 €	0,400 €
RK73B2BTTD121J	2	ud	R6, R7	Resistores de película gruesa - SMD 1/4watts 120ohms 5%	0,066 €	0,132 €
RK73B1JTTD103J	2	ud	R8, R21	Resistores de película gruesa - SMD 1/10WATT 10KOHMS	0,038 €	0,076 €
RK73B1JTTD2R7J	4	ud	R13, R14, R15, R16	Resistores de película gruesa - SMD 1/10watts 2.7ohms 5%	0,050 €	0,200 €
RK73B1JTTD221J	1	ud	R19	Resistores de película gruesa - SMD 1/10watts 220ohms 5%	0,050 €	0,050 €
CR0603-FX-1001HLF	2	ud	R20, R22	Resistores de película gruesa - SMD 1KOHM 1/10WATT 1%	0,033 €	0,066 €
CR0603-FX-1200GLF	3	ud	R23, R24, R25	Resistores de película gruesa - SMD 120OHM 1/10WATT 1%	0,033 €	0,099 €
CR0603-JW-222GLF	17	ud	R26, R27, R28, R29, R30, R34, R37, R42, R45, R50, R53, R58, R61, R68, R72, R75, R79	Resistores de película gruesa - SMD 2.2KOHM 1/10WATT 5%	0,050 €	0,850 €
RTD14005F	12	ud	RE1, RE2, RE3, RE4, RE5, RE6, RE7, RE8, RE9, RE10, RE11, RE12	Relés universales SP 16A 5mm 5VDC F	2,140 €	25,680 €
FR01FR16H-06XL	1	ud	SW1	Conmutadores giratorios codificados 16 POS REAL CODE FLUSH RA PC .2 SPAC	2,900 €	2,90 €
STM32F101V8T6	1	ud	U1	Microcontroladores ARM (MCU) 32BIT Cortex M3 64KB 20KB RAM 2X12 ADC	6,220 €	6,220 €
ST485ERBDR	4	ud	U2, U3, U5, U6	Circuito integrado para interfaz RS-422/RS-485 5V Hi Spd 30Mbps Lo Pwr Half Duplex	1,920 €	7,680 €
LM1117DT-3.3/NOPB	1	ud	U4	Regulador de voltaje - LDO 800MA LDO LINEAR REG	0,828 €	0,828 €
ULN2003V12DR	2	ud	U10, U13	Unidades y componentes periféricos (PCI) Low Pwr Relay Driver	0,228 €	0,456 €
ABM8G-12.000MHZ-18-D2Y-T	1	ud	Y1	Cristal 12.000MHz 18pF 30ppm -40C +85C	1,130 €	1,130 €
Phoenix 1792304	1	ud	-	Bloques terminales conectables 5.08 8P PLUG 90º	1,670 €	1,670 €
IND2002A	1	ud	-	Tarjeta de circuito impreso	111,430 €	111,430 €
3057 BK005	2	m	-	Cable de conexión 16AWG 26/30 PVC SPOOL BLACK	1,130 €	2,260 €
Total						**265,99** €

Tabla 5. Presupuesto del módulo *IND2002A*

IND5002						
Part Number del fabricante	Cantidad	Ud medida	Designador	Descripción	Precio/ud	Precio total
VJ0603V104ZXXCW1BC	18	ud	C1, C2, C3, C6, C7, C8, C9, C10, C12, C13, C14, C16, C17, C18, C19, C20, C21, C22	Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 0.1uF 25volts Y5V +80-20%	0,050 €	0,900 €
TAJB106K010RNJ	2	ud	C11, C15	Capacitadores de tantalio - SMD sólido 10volts 10uF 10%	0,107 €	0,214 €
VJ0603A120FXACW1BC	2	ud	C23, C24	Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 12pF 50volts C0G 1%	0,050 €	0,100 €
VESD05A1A-HD1-GS08	4	ud	D1, D2, D3, D4	Supresores de ESD 5.0 Volt 16 Amp	0,406 €	1,624 €
APA2106SURCK	1	ud	D5	LED estándar - SMD Hyper Red R.A. 635nm, 250mcd	0,085 €	0,085 €
APA2106SYCK	1	ud	D6	LED estándar - SMD Yellow 590nm Water Clear 150mcd	0,093 €	0,093 €
APA2106CGCK	1	ud	D7	LED estándar - SMD Green 570nm Water Clear 60mcd	0,101 €	0,101 €
86093487313H55ELF	1	ud	J2	Conectores DIN 41612 48P R/A PLUG DIN 41612	1,3 €	1,3 €
15-91-2100	1	ud	J3	Alojamientos de cables y cabecera C-GRID 10 CKT BKWY H	1,050 €	1,050 €
2011602101	1	ud	J5	Conectores DIN 41612 160P MALE R/A SOLDER	14,390 €	14,390 €
MPZ2012S101A	4	ud	L1, L2, L3, L4	Gránulos, chips y matrices de filtros EMI Ferrite Chip Beads	0,058 €	0,232 €
1755794	1	ud	P1	Bloques terminales conectables 5.08 8P HEADER 180DG	1,670 €	1,670 €
RK73B1JTTD472J	4	ud	R9, R10, R11, R12	Resistores de película gruesa - SMD 1/10watt 4.7Kohms 5%	0,050 €	0,200 €
RK73B1JTTD103J	2	ud	R8, R21	Resistores de película gruesa - SMD 1/10WATT 10KOHMS	0,038 €	0,076 €
RK73B1JTTD2R7J	4	ud	R13, R14, R15, R16	Resistores de película gruesa - SMD 1/10watts 2.7ohms 5%	0,050 €	0,200 €
RK73B1JTTD221J	1	ud	R19	Resistores de película gruesa - SMD 1/10watts 220ohms 5%	0,050 €	0,050 €
CR0603-FX-1001HLF	3	ud	R20, R22, R87	Resistores de película gruesa - SMD 1KOHM 1/10WATT 1%	0,033 €	0,099 €
CR0603-FX-1200GLF	3	ud	R23, R24, R25	Resistores de película gruesa - SMD 120OHM 1/10WATT 1%	0,033 €	0,099 €
CR0603-JW-222GLF	55	ud	R26, R27, R28, R29, R30, R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42, R43, R44, R45, R47, R48, R49, R50, R51, R52, R53, R55, R56, R57, R58, R59, R60, R61, R62, R63, R64, R65, R66, R67, R68, R69, R70, R71, R72, R73, R74, R75, R76, R77, R78, R79, R80, R81, R82	Resistores de película gruesa - SMD 2.2KOHM 1/10WATT 5%	0,050 €	2,750 €
EC2-5NJ	50	ud	RE1, RE2, RE3, RE4, RE5, RE6, RE7, RE8, RE9, RE10, RE11, RE12, RE13, RE14, RE15, RE16, RE17, RE18, RE19, RE20, RE21, RE22, RE23, RE24, RE25, RE26, RE27, RE28, RE29, RE30, RE31, RE32, RE33, RE34, RE35, RE36, RE37, RE38, RE39, RE40, RE41, RE42, RE43, RE44, RE45, RE46, RE47, RE48, RE49, RE50	Relés de señal baja - PCB 5VDC NON-LATCH	1,880 €	94,000 €
FR01FR16H-06XL	1	ud	SW1	Conmutadores giratorios codificados 16 POS REAL CODE FLUSH RA PC .2 SPAC	2,900 €	2,900 €
STM32F101V8T6	1	ud	U1	Microcontroladores ARM (MCU) 32BIT Cortex M3 64KB 20KB RAM 2X12 ADC	6,220 €	6,220 €
ST485ERBDR	2	ud	U5, U6	Circuito integrado para interfaz RS-422/RS-485 5V Hi Spd 30Mbps Lo Pwr Half Duplex	1,920 €	3,840 €
LM1117DT-3.3/NOPB	1	ud	U4	Regulador de voltaje - LDO 800MA LDO LINEAR REG	0,828 €	0,828 €
ULN2003V12DR	8	ud	U7, U8, U9, U10, U11, U12, U13, U14	Unidades y componentes periféricos (PCI) Low Pwr Relay Driver	0,228 €	1,824 €
ABM8G-12.000MHZ-18-D2Y-T	1	ud	Y1	Cristal 12.000MHz 18pF 30ppm -40C +85C	1,130 €	1,130 €
Phoenix 1792304	1	ud	-	Bloques terminales conectables 5.08 8P PLUG 90º	1,670 €	1,670 €
IND5002	1	ud	-	Tarjeta de circuito impreso	156,160 €	156,160 €
Total						**293,81 €**

Tabla 6. Presupuesto del módulo *IND5002*

IND5003						
Part Number del fabricante	Cantidad	Ud medida	Designador	Descripción	Precio/ud	Precio total
VJ0603V104ZXXCW1BC	18	ud	C1, C2, C3, C6, C7, C8, C9, C10, C12, C13, C14, C16, C17, C18, C19, C20, C21, C22	Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 0.1uF 25volts Y5V +80-20%	0,050 €	0,900 €
TAJB106K010RNJ	2	ud	C11, C15	Capacitadores de tantalio - SMD sólido 10volts 10uF 10%	0,107 €	0,214 €
VJ0603A120FXACW1BC	2	ud	C23, C24	Condensadores de cerámica multicapa (MLCC- SMD/SMT) 0603 12pF 50volts C0G 1%	0,050 €	0,100 €
VESD05A1A-HD1-GS08	4	ud	D1, D2, D3, D4	Supresores de ESD 5.0 Volt 16 Amp	0,406 €	1,624 €
APA2106SURCK	1	ud	D5	LED estándar - SMD Hyper Red R.A. 635nm, 250mcd	0,085 €	0,085 €
APA2106SYCK	1	ud	D6	LED estándar - SMD Yellow 590nm Water Clear 150mcd	0,093 €	0,093 €
APA2106CGCK	1	ud	D7	LED estándar - SMD Green 570nm Water Clear 60mcd	0,101 €	0,101 €
86093487313H55ELF	1	ud	J2	Conectores DIN 41612 48P R/A PLUG DIN 41612	1,300 €	1,300 €
15-91-2100	1	ud	J3	Alojamientos de cables y cabecera C-GRID 10 CKT BKWY H	1,050 €	1,050 €
2011602101	1	ud	J5	Conectores DIN 41612 160P MALE R/A SOLDER	14,390 €	14,390 €
MPZ2012S101A	4	ud	L1, L2, L3, L4	Gránulos, chips y matrices de filtros EMI Ferrite Chip Beads	0,058 €	0,232 €
1755794	1	ud	P1	Bloques terminales conectables 5.08 8P HEADER 180DG	1,670 €	1,670 €
RK73B1JTTD472J	4	ud	R9, R10, R11, R12	Resistores de película gruesa - SMD 1/10watt 4.7Kohms 5%	0,050 €	0,200 €
RK73B1JTTD103J	2	ud	R8, R21	Resistores de película gruesa - SMD 1/10WATT 10KOHMS	0,038 €	0,076 €
RK73B1JTTD2R7J	4	ud	R13, R14, R15, R16	Resistores de película gruesa - SMD 1/10watts 2.7ohms 5%	0,050 €	0,200 €
RK73B2BTTD121J	2	ud	R17, R18	Resistores de película gruesa - SMD 1/4watts 120ohms 5%	0,066 €	0,132 €
RK73B1JTTD221J	1	ud	R19	Resistores de película gruesa - SMD 1/10watts 220ohms 5%	0,050 €	0,050 €
CR0603-FX-1001HLF	3	ud	R20, R22, R83	Resistores de película gruesa - SMD 1KOHM 1/10WATT 1%	0,033 €	0,099 €
CR0603-FX-1200GLF	3	ud	R23, R24, R25	Resistores de película gruesa - SMD 120OHM 1/10WATT 1%	0,033 €	0,099 €
CR0603-JW-222GLF	57	ud	R26, R27, R28, R29, R30, R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42, R43, R44, R45, R46, R47, R48, R49, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59, R60, R61, R62, R63, R64, R65, R66, R67, R68, R69, R70, R71, R72, R73, R74, R75, R76, R77, R78, R79, R80, R81, R82	Resistores de película gruesa - SMD 2.2KOHM 1/10WATT 5%	0,050 €	2,850 €
EC2-5NJ	52	ud	RE1, RE2, RE3, RE4, RE5, RE6, RE7, RE8, RE9, RE10, RE11, RE12, RE13, RE14, RE15, RE16, RE17, RE18, RE19, RE20, RE21, RE22, RE23, RE24, RE25, RE26, RE27, RE28, RE29, RE30, RE31, RE32, RE33, RE34, RE35, RE36, RE37, RE38, RE39, RE40, RE41, RE42, RE43, RE44, RE45, RE46, RE47, RE48, RE49, RE50, RE51, RE52	Relés de señal baja - PCB 5VDC NON-LATCH	1,880 €	97,760 €
FR01FR16H-06XL	1	ud	SW1	Conmutadores giratorios codificados 16 POS REAL CODE FLUSH RA PC .2 SPAC	2,900 €	2,900 €
STM32F101V8T6	1	ud	U1	Microcontroladores ARM (MCU) 32BIT Cortex M3 64KB 20KB RAM 2X12 ADC	6,220 €	6,220 €
ST485ERBDR	2	ud	U5, U6	Circuito integrado para interfaz RS-422/RS-485 5V Hi Spd 30Mbps Lo Pwr Half Duplex	1,920 €	3,840 €
LM1117DT-3.3/NOPB	1	ud	U4	Regulador de voltaje - LDO 800MA LDO LINEAR REG	0,828 €	0,828 €
ULN2003V12DR	8	ud	U7, U8, U9, U10, U11, U12, U13, U14	Unidades y componentes periféricos (PCI) Low Pwr Relay Driver	0,228 €	1,824 €
ABM8G-12.000MHZ-18-D2Y-T	1	ud	Y1	Cristal 12.000MHz 18pF 30ppm -40C +85C	1,130 €	1,130 €
Phoenix 1792304	1	ud	-	Bloques terminales conectables 5.08 8P PLUG 90º	1,670 €	1,670 €
IND5003	1	ud	-	Tarjeta de circuito impreso	156,160 €	156,160 €
Total						**297,8 €**

Tabla 7. Presupuesto del módulo *IND5003*

Si bien puede parecer un precio excesivo para su desarrollo e implementación, hay que advertir que los presupuestos se encuentran dirigidos hacia la fabricación de prototipos y disminuyen enormemente frente a la producción de múltiples módulos de relés, especialmente en la elaboración de las tarjetas de circuito impreso.

El importe de los artilugios que han intervenido en la evolución del sistema también son tomados en cuenta, pero conforman una pequeña parte del total del coste de los recursos *hardware*, tal y como muestra la tabla 8.

Part Number	Cantidad	Ud medida	Descripción	Precio/ud	Precio total
ST-LINK	1	ud	Depurador y grabador para la familia de controladores STM8 y STM32	17,00 €	17,00 €
84281210030134	1	ud	Alojamiento de cable y cabecera SOCKET	0,75 €	0,75 €
3365/10-100SF	1	m	Cables planos 10/CAB/RC/TYP1/ 28AWG/STR/.050"/100'	1,22 €	1,22 €
961220-6404-AR	1	ud	Alojamiento de cable y cabecera 20P STRT 2 ROW GOLD 5.5MM MATING PIN	0,75 €	0,75 €
N010-010-GY	1	ud	Cable Ethernet / cable de red cruzado CAT5e 350MHz GRAY	4,30 €	4,30 €
Total					**24,02 €**

Tabla 8. Presupuesto de cableado

El coste total asciende a 881.62 €, un precio asequible que hace notar una gran diferencia con las alternativas del mercado, aún con posibilidad de mermar más ante una producción más numerosa.

6.2 Coste de los recursos *software*

Los recursos software juegan un papel fundamental en el desarrollo e implementación del software y han sido proporcionados en el entorno de trabajo gracias a su uso en otros proyectos; sin embargo, aunque no han representado ningún coste en este sistema conviene evaluar el precio de cada una de las licencias del *software* empleado, de cara a un análisis económico individual del sistema. La tabla 9 ofrece una visión del importe de las licencias de *software*.

Licencia de Producto	Cantidad	Ud medida	Descripción	Precio/ud	Precio total
LabWindows/CVI Base	1	ud	Sistema de Desarrollo Base para LabWindows/CVI para Windows, Incluye 1 Año de SSP	**1.260,00 €**	1.260,00 €
Altium Designer 14	1	ud	Software completo de diseño de tarjetas de circuito impreso, capacidades de fabricación, verificación de sistemas	5.395,00 €	5.395,00 €
MDK-ARM	1	ud	Kit de desarrollo de microcontroladores MDK-ARM Microcontroller - Standard Edition (C/C++ Compiler + Assembler + µVision IDE/Debugger/Simulator + RTX Kernel)	4.091,10 €	4.091,10 €
NI-VISA 777300-00	1	ud	Utilización de la Arquitectura de Software de Instrumentos Virtuales	123,00 €	123,00 €
Total					**10.869,10 €**

Tabla 9. Presupuesto de licencias de software

6.3 Coste de los recursos humanos

En el proyecto, el recurso humano es únicamente su autor, y aunque desempeña labores muy diversas que ofrecerían diferentes costes asociados a su trabajo, se unifican todas las fases bajo el mismo coste horario hacia la empresa, para lo que se ha tenido en cuenta el sueldo bruto del trabajador y las cotizaciones a la seguridad social (aproximadamente el 31% distribuido en contingencias comunes, desempleo, formación profesional, fondo de garantía social y accidentes de trabajo y enfermedades profesionales).

El coste bruto anual hacia la compañía es de unos 28580 € por unas 1760 horas, lo que resulta en un importe por hora del trabajador de 16.24 €. Así, es posible determinar el valor agregado de cada una de las fases del proyecto, tal y como muestra la tabla 10.

Etapa de desarrollo		Cantidad (h)	Precio/h	Importe total
Diseño del Hardware				
	Análisis del sistema	28	16,24 €	454,72 €
	Elaboración de esquemáticos	92	16,24 €	1.494,08 €
	Diseño de las PCB	123	16,24 €	1.997,52 €
Diseño del Software				
	Análisis del sistema	16	16,24 €	259,84 €
	Diseño de la capa del firmware	131	16,24 €	2.127,44 €
	Diseño de la capa del driver	48	16,24 €	779,52 €
	Diseño de la capa de la GUI	82	16,24 €	1.331,68 €
	Simulaciones y depuraciones	37	16,24 €	600,88 €
Implantación y ajustes		29	16,24 €	470,96 €
Pruebas		44	16,24 €	714,56 €
Total		630	16,24 €	**10.231,20 €**

Tabla 10. Presupuesto de los recursos humanos

6.4 Coste total

El coste total de todos los recursos determina el importe económico del objeto del proyecto, lo que permite figurar el lapso de tiempo en el que se podría amortizar la inversión en el desarrollo del sistema de *switching* modular, aunque no se profundiza en ello. La tabla 11 engloba el presupuesto de los recursos empleados, lo que eleva el precio total del proyecto a 21981.92 €.

Recursos	Presupuesto
Recursos Hardware	881,62 €
Recursos Software	10.869,10 €
Recursos Humanos	10.231,20 €
Total	21.981,92 €

Tabla 11. Presupuesto total

Capítulo 7. Conclusiones y trabajo futuro

7.1 Conclusiones

El coste total del desarrollo parece excesivo, pero dado el importe de instrumentos de *switching* del mercado actual, representa una cifra asumible y amortizable en poco tiempo. Si bien, la mayoría de desarrolladores de dispositivos *VXI* gozan de pocos competidores y por tanto ejercen un oligopolio que les permite elevar ligeramente los precios, el desarrollo propio de instrumentos *VXI* resulta una opción muy idónea dada la situación actual.

A nivel técnico, los *microcontroladores* ofrecen una alternativa de diseño ideal para aplicaciones simples, como la de este proyecto, sin implicar unos costes de tiempo de desarrollo altos o un gran valor del dispositivo como comprometen otras tecnologías. La compatibilidad con un diseño de lenguaje de nivel alto, unido a la documentación adecuada (hojas de características y manuales de referencia) convierten la implementación en una labor con relativamente poca especialización en este tipo de tecnología.

Sin duda, el *hardware* erige un camino hacia el logro de los requisitos, pero también abre multitud de posibilidades ante posteriores evoluciones del sistema, lo que le confiere una gran versatilidad al sistema.

Las capas de *software* establecen en conjunto una estructura óptima que facilitará el trabajo futuro en diversas aplicaciones, aún más si ninguno de los desarrolladores conoce la implementación; sin embargo, individualmente ofrecen una completa operatividad aprovechando al máximo las prestaciones. El *firmware* se ha implementado de tal manera que permite una fácil incorporación de otros tipos de módulos de relés y una cómoda inclusión de posibles comandos adicionales que aumenten la funcionalidad, confiriendo mayor adaptabilidad. El *driver* aprovecha las capacidades que le brinda la capa del *firmware* y dota al usuario de pleno control sobre toda la utilidad del sistema con apenas unas cuantas funciones concisas por medio de *VISA*. Las funciones conceden una integración sencilla en entornos de desarrollo de *ATLAS* (*Abbreviated Test Language for All Systems*), como *PAWS* (*Professional ATLAS Workstation*), la herramienta mayoritaria en el desarrollo de conjuntos de programas de test de aviónica militar. La interfaz gráfica de usuario hace especial hincapié en la simplicidad, flexibilidad, agilidad, perceptibilidad y aclaración de información, de manera que el usuario experimente un dominio intuitivo con un gran rendimiento y numerosos servicios, con vistas a futuras transformaciones.

Los resultados del presente proyecto confirman una viabilidad técnica sencilla, que deriva de un análisis exhaustivo de las opciones disponibles y un estudio en profundidad de las elecciones. Además demuestran el estricto cumplimiento de los requisitos, como primera meta a lograr, mediante un diseño eficaz y eficiente.

Aunque la elaboración del proyecto y el sistema es satisfactoria, durante su consecución se presentaron diversos problemas que dificultaron alcanzar la culminación del proyecto, algunas de las decisiones tomadas se manifestaban adecuadas y se tornaron impedimentos. Por

ejemplo, la soldadura de los componentes *SMD* no albergaba gran complicación, pero aparecieron cortocircuitos en los elementos con una implantación más compleja (*microcontrolador*), que a simple vista eran imperceptibles, por lo que resultó en una ligera prolongación de la fase de las pruebas hasta solucionar las trabas. Aunque los requisitos establecen de forma clara los objetivos del proyecto, la forma de lograrlos deriva subjetivamente en un mar de contratiempos o en un camino confortable; dado que el *driver* representa la funcionalidad del sistema de cara al usuario, podría constituirse como un único instrumento virtual bajo la comunicación la interfaz *Ethernet-Serie*, con múltiples módulos de relés o como módulos de relés individuales, por lo que en un comienzo se consideró construir un solo instrumento virtual configurable en cuanto a módulos, pero tras ahondar en la idea se revelaron cuantiosas dificultades que redirigieron la constitución del controlador hacia una competencia individual de cada módulo de relés.

El proyecto ha contribuido significativamente al estudio y desarrollo de aplicaciones *hardware* y *software* partiendo desde cero, interviniendo en cada una de las etapas que conlleva una implementación exitosa del sistema, lo que se presenta como un episodio de motivación hacia la elaboración de dispositivos electrónicos.

Si bien parece que la implementación de instrumentos llega a su fin con la consecución de objetivos, siempre existen resquicios de funcionalidad añadida o de mejora complementarios y no necesarios, que inciden en un camino interminable hacia la perfección con los que el propio autor disfruta proyectando. Los dispositivos electrónicos constituyen obras que de un cierto modo albergan el estilo de proyección y trabajo de su creador y que por tanto, conforman una parte de él mismo.

La cantidad de técnicas y conceptos asimilados infiere en una experiencia gratificante y enriquecedora, que evidencia un progreso de utilidad en la carrera profesional y una ligera denotación en la dedicación técnica.

7.2 Trabajo futuro

El proyecto se centra principalmente en la implementación de tres módulos de relés distintos, con el respectivo software (*firmware, driver* y *GUI*) que haga posible su control, no obstante de cara a un uso pleno como instrumento *VXI*, requiere de la construcción de un *carrier* apropiado, que aloje una placa de circuito impreso (*backplane*) que facilite una comunicación con todos los módulos y suministre una alimentación proveniente del *backplane VXI,* a través de los conectores complementarios a los empleados en la parte trasera de los módulos de relés. Una alternativa frente al *VXI* se plantea desde los requisitos, la elaboración de un *mainframe* que albergue los módulos de relés y se apropie del bloque *Ethernet-RS485*, de forma que a través de una mecánica que contenga este bloque funcional y la alimentación adecuada, permita hospedar hasta 32 módulos de relés en ranuras dedicadas a uno o varios módulos superpuestos.

En la misma línea de diseño del sistema, existen numerosas tarjetas de relés con diferentes características y configuraciones que necesitan de la proyección adecuada, pero dada la naturaleza flexible del sistema, resulta una tarea muy asequible y sencilla. Cabe destacar que las tarjetas de relés de potencia del mercado actual presentan impedimentos en el comportamiento (pésima conmutación del relé o una alta impedancia de cortocircuito) frente a señales para las que no han sido proyectados, de poca intensidad, es por ello que convendría dedicar especial atención a la solución de este problema, mediante por ejemplo, el empleo de relés de baja potencia en paralelo y sus respectivas acomodaciones.

Por otra parte, profundizando en la propia implementación, aún es posible incrementar las funcionalidades que ofrece el instrumento, bien vía *software* o en su lugar a través del *hardware*, como un almacenamiento de errores a través del uso de registros en el microcontrolador o una detección automática de relés averiados.

Más allá del diseño de instrumentos dedicados a la conmutación de relés, hay multitud de instrumentos con cometidos muy diversos que podrían ser implementados, tanto si se opta por una elaboración *VXI* como si no, destinados a pruebas y mediciones de aviónica militar o designados a un ámbito más comercial.

Bibliografía

1. Abracon Corporation 2011, 04/20-last update, *ABM8G*. Available: http://www.abracon.com/Resonators/ABM8G.pdf [2013, 06/10].
2. Araya, G. 2011, 07/19/2011-last update, *Análisis y diseño de sistemas - Modelos para el desarrollo de software*. Available: http://osc.co.cr/analisis-y-diseno-de-sistemas-modelos-para-el-desarrollo-de-software/ [2013, 10/08].
3. ARM Holdings 2013, *Serial Wire Debug*. Available: http://www.arm.com/products/system-ip/debug-trace/coresight-soc-components/serial-wire-debug.php [2013, 11/10].
4. Brad 2006, 01/31/2006-last update, *PCB Trace Width Calculator*. Available: http://circuitcalculator.com/wordpress/2006/01/31/pcb-trace-width-calculator/ [2013, 11/15].
5. FCI 2011, 05/14-last update, *DIN STANDARD HEADER DIN 41612 STYLE C/2*. Available: http://portal.fciconnect.com/Comergent//fci/drawing/c-8609-2003.pdf [2013, 06/02].
6. Gerónimo Castillo, G. 2005, *Ethernet y Protocolos TCP/IPv4*. Available: http://mixteco.utm.mx/~resdi/historial/materias/IPv4.pdf [2013, 08/12].
7. Greenberg, C. 2007, 08/08-last update, *The VXI Bus: Well-Conceived for Demanding Test Applications*. Available: http://www.vxibus.org/q307newsletter.html [2013, 12/15].
8. Günther Griding, B.W. 2007, 02/26/2007-last update, *Introduction to Microcontrollers*. Available: http://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf [2013, 05/04].
9. HARTING Electronics GmbH & Co. 1996, 09/10-last update, *har-bus 64 male connector*. Available: http://datasheet.octopart.com/2011602101-HARTING-Elektronik-datasheet-14433910.pdf [2013, 05/17].
10. IEEE Computer Society 2012, *IEEE Standard for Ethernet*, New York, USA.
11. IVI Foundation 2013, 11/12-last update, *IVI-3.1: Driver Architecture Specification*. Available: http://www.ivifoundation.org/downloads/Architecture%20Specifications/IVI-3.1_Architecture_2013-11-12.pdf [2013, 06/15].
12. IVI Foundation 2013, 07/26-last update, *IVI-3.2: Inherent Capabilities Specification*. Available: http://www.ivifoundation.org/downloads/Architecture%20Specifications/IVI-3.2_Inherent_Capabilities_2013-07-26.pdf [2013, 06/31].
13. Kingbright 2013, 03/12-last update, *21x0.6mm RIGHT ANGLE SURFACE LED LAMP*. Available: http://www.kingbrightusa.com/images/catalog/spec/APA2106SURCK.pdf [2013, 05/29].
14. Lantronix 2013, 02/15-last update, *XPort Data Sheet*. Available: http://www.lantronix.com/pdf/XPort_DS.pdf [2013, 06/04].
15. Lantronix 2013, 02/01/2013-last update, *XPort User Guide*. Available: http://www.lantronix.com/pdf/XPort_UG.pdf [2013, 06/07].
16. Mandado Pérez, E. & Mandado Rodríguez, Y. 2007, "Tecnologías de Implementación de los circuitos digitales" in *Sistemas electrónicos digitales*, Novena edn, MARCOMBO, S.A., Barcelona.
17. Martinez, M. 2001, *Principios de diseño de las interfaces gráficas* [Homepage of URJC], Available: http://www.escet.urjc.es/~intgraf/documentos/Principios-Disenno-Delphi.pdf [2013, 11/03].

18. Molex 2013, 12/26-last update, *15-91-2100*. Available: http://www.molex.com/webdocs/datasheets/pdf/en-us/0015912100_PCB_HEADERS.pdf [2013, 05/14].
19. National Instruments Corporation 2013, 08/05/2013-last update, *NI-VISA 5.4*. Available: http://www.ni.com/download/ni-visa-5.4/4230/en/ [2013, 09/29].
20. National Instruments Corporation 2012, *National Instruments VISA*. Available: http://www.ni.com/visa/ [2013, 10/12].
21. National Instruments Corporation 2012, 02/02/2012-last update, *What is an Intrument Driver?* Available: http://www.ni.com/white-paper/4803/en/ [2013, 10/19].
22. National Instruments Corporation 2010, 11/08/2010-last update, *Short Tutorial on VXI*. Available: http://www.ni.com/white-paper/2899/en/ [2013, 10/15].
23. National Instruments Corporation 2006, 06/06/2006-last update, *Comunicación Serial: Conceptos Generales*. Available: http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1 [2013, 10/22].
24. National Instruments Corporation 2003, *LabWindows/CVI Instrument Driver Developers Guide*. Available: http://www.ni.com/pdf/manuals/370699a.pdf [2013, 10/07].
25. National Instruments Corporation 1996, 07/06-last update, *NI-VXI User Manual*. Available: http://www.slac.stanford.edu/grp/cd/soft/vxworks/doc/cpu/vxi/68k/nicpu030/NI-VXIUsersMan.pdf [2013, 07/13].
26. NEC Corporation 2006, 03/01-last update, *Miniature Signal Relay EC2 SERIES*. Available: http://www.worldproducts.com/pdfs/ec2.pdf [2013, 05/18].
27. NKK Switches 2013, *Series FR01*. Available: http://www.nkkswitches.com/pdf/FR01.pdf [2013, 05/18].
28. Phoenix Contact 2013, 02/07-last update, *Base strip- MSTBVA 2,5/ 8-G-5,08 - 175794*. Available: https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail%3Apid=1755794&library=usen&pdfmode=direct&pdflanguage=en [2013, 06/02].
29. Phoenix Contact 2013, 02/07/2013-last update, *Printed-circuit board connector - MVSTBE 2,5/ 8-ST-5,08 - 1792304*. Available: http://www.phoenixcontact.com/us/products/1792304/pdf[2013, 07/04].
30. Pizzica, S. 2001, *Open Systems Architecture Solutions for Military Avionics Testing*, El segundo, CA, USA.
31. Positronic Industries 2013, *Standard Density Regular Connectors*. Available: http://www.connectpositronic.com/pdf_view/49/c009revd_stddenrec.pdf [2013, 05/20].
32. Riu Costa, P.J., Rosell Ferrer, J. & Ramos Castro, J. 2000, *Sistemas basados en el bus VME y VXI* [Homepage of Ediciones UPC]. Available: http://melca.com.ar/archivos/apuntes/Sistemas%20de%20instrumentacion/EE03702C.pdf [2013, 09/15].
33. Schweers, R.J. 2003, *Metodologías de diseño de hardware*, Universidad Nacional de la Plata.
34. Sebastian Smith, M.J. 1997, "Types of ASICs" in *Applicantion-Specific Integrated Circuits*, 1ª edn, Addison Wesley Publishing Company, Reading, Massachusetts, pp. 1040.
35. STMicroelectronics 2013, *STM32F101*. Available: http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN1567 [2013, 05/04].
36. STMicroelectronics 2013, 04/20-last update, *STM32F101x8*. Available: http://www.st.com/st-web-

ui/static/active/en/resource/technical/document/datasheet/CD00161561.pdf [2013, 05/03].

37. STMicroelectronics 2011, *RM0008 Reference Manual*. Available: http://www.st.com/web/en/resource/technical/document/reference_manual/CD00171190.pdf [2013, 07/05].
38. STMicroelectronics 2011, *UM0627 User Manual*. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/CD00221563.pdf?s_searchtype=keyword [2013, 09/09].
39. STMicroelectronics 2006, 10/04-last update, *ST485ERBDR Datasheet*. Available: http://pdf1.alldatasheet.es/datasheet-pdf/view/186209/STMICROELECTRONICS/ST485ERBDR.html [2013, 06/04].
40. TDK 2013, 12/04-last update, *Chip Beads*. Available: http://product.tdk.com/emc/beads/en/documents/beads_commercial_power_mpz_en.pdf [2013, 05/30].
41. Texas Instruments 2012, 05/03-last update, *ULN2003V12*. Available: http://www.ti.com/lit/ds/slrs060b/slrs060b.pdf [2013, 05/29].
42. Texas Instruments 2011, 10/01/2011-last update, *Choosing an Appropiate Pull-up/Pull-Down Resistor for Open Drain Outputs*. Available: http://www.ti.com/lit/an/slva485/slva485.pdf [2013, 11/16].
43. Texas Instruments 2000, 02/10-last update, *LM1117-N, LM1117*|. Available: http://www.ti.com/lit/ds/snos412m/snos412m.pdf [2013, 05/30].
44. Tognazzini, B. 2003, *First Principles of Interaction Design*. Available: http://www.asktog.com/basics/firstPrinciples.html [2013, 11/01].
45. Tyco Electronics Corporation 2011, 04/15-last update, *Power PCB Relay RT1*. Available: http://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=RT1pb&DocType=DS&DocLang=EN [2013, 06/05].
46. Vega Luna, J.I., Sánchez Gonzalez, R., Salgado Guzmán, G. & Sánchez González, L.A. 1996, 09/09/1996-last update, *Arquitectura RISC vs CISC* [Homepage of Ed. Prentice Hall.]. Available: http://www.azc.uam.mx/publicaciones/enlinea2/num1/1-2.htm [2013, 10/25].
47. Vishay 2012, 06/20-last update, *VESD05A1A-HD1*. Available: http://www.vishay.com/docs/81800/vesd05a1.pdf [2013, 07/05].
48. VTI Instruments Corporation 2012, ***VXI as an Established Bus***. Available: http://www.vtiinstruments.com/VXI-Established-Bus.aspx [2013, 09/02].
49. VXI Plu&Play Systems Alliance 2012, 12/11-last update, *VPP-4.3: The VISA Library*. Available: http://www.ivifoundation.org/docs/vpp43%20and%20vpp432%20Oct%202012/vpp43.pdf [2013, 05/15].
50. VXI Plug&Play Systems Alliance 2013, 03/06-last update, *VPP-1: Charter Document*. Available: http://www.ivifoundation.org/docs/vpp1_2013-03-06.pdf [2013, 06/04].
51. VXI Plug&Play Systems Alliance 2013, 06/03-last update, *VPP-2: System Frameworks Specification*. Available: http://www.ivifoundation.org/docs/vpp2_2013-03-06.pdf [2013, 05/06].
52. VXI Plug&Play Systems Alliance 2008, 04/14-last update, *VPP-3.1: Instrument Drivers Architecture and Design Specification*. Available: http://www.ivifoundation.org/docs/vpp31.pdf [2013, 06/03].
53. VXI Plug&Play Systems Alliance 2008, 04/14-last update, *VPP-3.2: Instrument Driver Functional Body Specification*. Available: http://www.ivifoundation.org/docs/vpp32.pdf [2013, 06/07].

54. VXI Plug&Play Systems Alliance 2008, 04/14-last update, *VPP-3.4 Instrument Driver Programmatic Developer Interface Specification*. Available: http://www.ivifoundation.org/docs/vpp34.pdf [2013, 06/12].
55. VXIbus Consortium Inc. 2010, 05/06-last update, *VXIbus System Specification*. Available: http://www.vxibus.org/files/VXI_Specs/VXI-1_4-0%2020100527.pdf [2013, 06/19].
56. Yiu, J. 2013, "The definitive guide to the ARM CORTEX-M3" in, 2ª edn, Elsevier Inc., Burlington, USA.
57. Yoon, T., Chloe, T. & C Hong, J. Park 2004, 11/12/2004-last update, *Implementation of a time-frequency domain reflectometry system with PXI platform for a coaxial cable.* Available: http://ece.utah.edu/~cfurse/Center%20of%20Excellence/wiring_papers/Implementation_of_a_time_frequency_domain_reflectometry_system_with%20PXI_platform_for_a_coaxial_cable_Tokson_may2004.pdf [2013, 04/24].

Anexos

A.1 Sistemas *VXI*

A.1.1 Historia de los sistemas *VXI*

Tal y como se detalla en la base de datos de *VTI Instruments Corporation* [48], la historia de los estándares más populares (*GPIB, VXI, PXI y LXI)*, se remonta más de 30 años en el tiempo.

IEEE estandarizó el *GPIB* en 1975. Desarrollado en los últimos años de la década de los 60 por un consorcio de empresas (inicialmente *Hewlett- Packard, Tektronix, Racal-Dana, Wavetek, y Colorado Data Systems*, y más tarde se unieron *National Instruments, Bruel & Kjaer, Keithley, Fluke, y Genrad*), *GPIB* fue diseñado para propiciar una comunicación más fácil entre instrumentos y controladores. Desde que fue aceptado como *bus* ampliamente utilizado en los primeros años de la década de los 70, *GPIB* ha sido incorporado en miles de instrumentos y continúa siendo incluido en nuevos diseños de instrumentos.

Durante años los fabricantes de instrumentación han introducido tecnologías destinadas a suplantar el *GPIB*, pero estas tecnologías carecieron de la aceptación de la industria y entonces desaparecieron rápidamente porque obligaron al cliente a ceñirse únicamente a uno o dos suministradores. Los sistemas de integración necesitan la flexibilidad de una solución de múltiples vendedores para mantener la tecnología actualizada, un precio competitivo y tomar ventaja de toda una industria ofertando productos en lugar de una o dos compañías.

En los últimos 20 años, los proyectos aeroespaciales y militares de gran duración han encontrado una solución a sus problemas en el *VXIbus*. *VXI* tiene una larga historia como estándar abierto industrial, continuamente innovando y creciendo con gran popularidad. En 1989, el *MXIbus* proporcionaba soluciones a control remoto con una gran capacidad y a bajo coste para los sistemas *VXI*. Esto, unido al desarrollo de la tecnología *PCI Express* cableada para el control remoto del *VXI* asegura que la plataforma seguirá vigente a lo largo de muchos años.

El segundo estándar abierto de la industria para sistemas de instrumentación fue el *VXIbus*, que surgió principalmente para superar las limitaciones del *GPIB* y promover un estándar no solo dedicado a la prueba y medición, sino para cualquier industria. Las especificaciones del *VXI* fueron definidas en 1987 y aceptadas como industria estándar por *IEEE* en 1993. La definición de las especificaciones del *VXI* conducida por el Departamento de Defensa de Estados Unidos y su necesidad de reducir el tamaño físico de los *racks* de sistemas de instrumentación, creó una ajustada cadencia y sincronización entre múltiples instrumentos, y logra unos índices de transferencia más rápidos de los que el *GPIB* puede ofrecer.

En *Sistemas basados en el bus VME y VXI de* Riu Costa, P.J., Rosell Ferrer, J. y Ramos Castro, J. [32], se explaya el origen y concepción del *VXIbus*.

El *VXIbus* surgió bajo la influencia de las especificaciones del *bus VME (VersaModular Eurocard)*. El *VMEbus* es un estándar de *bus* informático, diseñado originalmente para los microprocesadores *Motorola 68000*, pero posteriormente se comenzó a usar en multitud de

aplicaciones y se estandarizó como el *ANSI/IEEE 1014-1987* por *IEC*. En términos de *hardware* se fundamenta en los formatos *Eurocard* (una tarjeta de circuito impreso europea que usa conector de 96 pines), los elementos mecánicos y conectores *DIN 41612*; sin embargo, emplea su propia configuración de conexiones (patillaje), ya que *Eurocard* no lo especifica. Aunque nació en 1981 aún es considerablemente usado.

En los orígenes de la instrumentación modular existían tarjetas que contenían toda la funcionalidad que existía, pero pronto la industria comenzó a demandar mayor funcionalidad de la que podría caber en una sola tarjeta. Así, los diseñadores de tarjetas se percataron rápidamente de que podrían continuar sus proyectos en tres dimensiones, por lo que nació el concepto de *mezzanine*. Un módulo *mezzanine* cumple posee un tamaño reducido que le permiten establecerse en la base de los instrumentos y encajar perfectamente entre otros módulos en el sistema de *bus*, por lo que permiten añadir funcionalidad sin malgastar ranuras adicionales del *bus*. En un principio los módulos *mezzanine* se manufacturaban principalmente por fabricantes de *VMEbus*, debido especialmente a dos razones:

- Cuando se produjo la transición a 32bits, el espacio limitado en una tarjeta *VME* no permitía incorporar toda la funcionalidad que requería la aplicación.
- Satisfacer la demanda del cliente por diferentes tipos de entradas y salidas en la misma base de la tarjeta.

A la economía del fabricante le complace la producción a gran escala de un solo tipo de módulo, pero el consumidor necesita diferentes funcionalidades para distintas aplicaciones. De esta manera, se le ocurrió a un grupo de fabricantes producir un módulo base con *CPU* y entonces utilizar los módulos *mezzanine* para personalizar la tarjeta para las diversas funcionalidades.

Aunque la idea *mezzanine* había añadido más flexibilidad al *VMEbus*, lo había hecho a costa de la compatibilidad, porque no había un estándar *mezzanine*, cada fabricante había desarrollado su propia arquitectura de *bus*. Muchos simplemente utilizaron una variante del *bus* del microprocesador 68000 desde que era el *bus* local sobre el que se había basado su diseño. Como resultado, ninguna de las tarjetas *mezzanine* entre los cientos de fabricantes de *VME* es compatible aunque cubriesen las mismas necesidades de funcionalidad. A finales de la década de los 80 *VITA* trató de establecer un consenso por un *mezzanine* estándar común sin ningún éxito. Los fabricantes consideraban que ser propietario de *bus*es *mezzanin*e distintos sería una manera de diferenciar sus productos.

Desde entonces han ocurrido diversos acontecimientos. En primer lugar, *Greensprings* introdujo *IndustryPack* en el mercado como un *mezzanine* estándar en la industria para funciones de entradas y salidas. Inicialmente, la idea no fue bien acogida por los fabricantes de *VME*, pero recibió un gran apoyo de *Motorola* en 1992 cuando esta introdujo su tarjeta única *MVME 162* para ordenador con cuatro ranuras para los módulos de *IndustryPack*. Más tarde, a raíz del lanzamiento de la *MVME 162* otros comerciantes de *VME* desarrollaron sus productos de modo que aceptarán los productos de *IndustryPack*.

En Europa, casi al mismo tiempo, *MEM Mikro Elektronik Gmbh* introdujo el concepto de *M-Modules*. Durante 1990 y 1991 *MEM* y otros fabricantes europeos fundaron *MUMM*

(*Manufacturers and users of M-Module*s), una asociación independiente para promover el progreso y el uso de los *M-Modules* en el mundo.

En segundo lugar, comenzó un segundo intento por una especificación de tarjeta *mezzanine* común (*CMC*). Conocida como *P1386*, este esfuerzo por tratar de definir unas especificaciones comunes y un mismo formato de conector para los módulos *mezzanine* fue usado por *VMEbus, Futurebus+, Multibus II*, y otros módulos embebidos a tiempo real. Ahondando en *P1386*, *P1386.1* denominado *PMC,* define el *bus PCI* en el factor de forma de *CMC* y *P1386.2,* denominado *SMC*, define el *S-Bus* en el factor de forma de *CMC*.

Gracias a su gran aceptación por ser el mayor fabricante de semiconductores, *PCI*, comenzó a ser la elección por defecto de *bus* local de muchos diseñadores de placas de microprocesadores.

VME proyectó su *bus* de manera que la transmisión es asíncrona, es decir, que la transferencia no está ligada a un reloj, las tarjetas son *Direct memory Access (DMA)* y cada tarjeta es maestra o esclava. El *bus* asíncrono se divide en cuatro sub-*bus*es:

- En el *sub-bus* de transferencia de datos *VME* usa por separado, dos *bus*es de 32 líneas de datos y 32 líneas de dirección. El direccionamiento del *bus 68000* era en realidad de 24 bits y el *bus* de datos de 16 bits (aunque internamente fuese 32bits/32bits), pero el diseño de *VME bus*caba implementaciones usando los 32bits. Para permitir ambos anchos de *bus*, *VME* usa dos conectores de *Eurocard* diferentes.
- El *sub-bus* de arbitraje de transferencia de datos es controlado por un conjunto de nueve líneas, que posibilita el traspaso del control del *bus* de datos entre los diferentes elementos maestro de manera estructurada y asegurando que solo una tarjeta maestra controla el *bus* en cualquier momento. Todas las comunicaciones son controladas por la tarjeta en la primera ranura del chasis *Eurocard*, nombrado como módulo árbitro o controlador. Hay dos formas de controlar las tarjetas, priorizando y seleccionando todos los elementos de manera equitativa y en un orden racional (*Round Robin*), generalmente empezando desde el primer elemento hasta el último. Independientemente del modo de control, una tarjeta también puede convertirse en maestra colocándola en una de las cuatro primeras ranuras y según el método de arbitraje seleccionará cual será la tarjeta controladora, y de esta manera adquirir el acceso al *bus*. En la transmisión de datos por parte de cualquier tarjeta, esta determina la dirección (a la que llegarán los datos), un modificador de dirección (utilizados para dividir el espacio de direccionamiento del *bus VME* en distintos subespacios) y datos en el *bus*. En la lectura de datos el recurso es esencialmente el mismo, únicamente la tarjeta controladora determina la dirección e indica al *bus* que debe permanecer en un *tri-state* (un estado de alta impedancia).
- El *sub-bus* de interrupciones dispone de siete líneas de interrupción priorizadas y hasta siete controladores de interrupciones, que proporciona a los dispositivos la capacidad de realizar sus peticiones de interrupción y que sean comprobadas por el elemento maestro del sistema.
- El *sub-bus* de utilidades abastece la alimentación, la detección de fallos en dicha alimentación, señales de reloj, señales de inicialización del sistema y una línea auxiliar de transferencia de datos serie.

No obstante, el *bus VME* no estaba orientado al diseño de sistemas de instrumentación, ya que no integraba líneas analógicas, ni de sincronización.

La arquitectura del *VXI bus* es una plataforma de estándar abierta para test automatizados basados en *VMEbus*. *VXI* continúa las extensiones de instrumentación de *VME*, definiendo líneas de *bus* adicionales para la cadencia y disparo, así como para necesidades mecánicas y protocolos estándar para configuración, comunicación basada en mensajes, extensión múltiple en chasis y otras características. El *bus* de *VXI* añade siete sub-*bus*es más a los cuatro sub-*bus*es de *VME*:

- El *sub-bus* de reloj está constituido por dos señales de reloj de 100MHz y 10 MHz y una señal de sincronización de reloj para la señal de reloj de 100 MHz, todas señales diferenciales con niveles *ECL*. La controladora del *bus VXI* puede producir dichos relojes o pueden ser proporcionados de manera externa.
- El *sub-bus* en estrella está formado por línea bidireccionales diferenciales con niveles *ECL*, que otorgan una vía de comunicación entre dispositivos para aplicaciones de temporización decisiva.
- El *sub-bus* de disparo ofrece ocho líneas de disparo con niveles *TTL* y seis líneas con niveles *ECL*, que permiten el desarrollo de complicadas pautas de temporización y disparo entre instrumentos, y la comunicación mediante el agrupamiento de líneas, que complementan al *bus VME.*
- El *sub-bus* local está integrado por 72 líneas que comunican a cada dispositivo con sus inmediatamente adyacentes, de tal manera que relaciona localmente diferentes instrumentos a alta velocidad.
- El *sub-bus* de suma analógica constituye un nodo común a todos los módulos del sistema. Cada elemento del sistema puede transmitir a esta línea a través de una fuente de corriente con una impedancia superior a 10kG y una capacidad en paralelo menor de 4pF. El receptor deberá tener una capacidad en paralelo más restrictiva, de 3pF.
- El *sub-bus* de identificación de módulos está constituido por 12 líneas conectadas a cada una de las 12 ranuras en el *bus VXI* para el conexionado de tarjetas. Estas líneas se alimentan del dispositivo conectado a la ranura 0 y posibilitan la detección de instrumentos conectados al *bus*. El dispositivo conectado a la ranura 0 debe ser un instrumento con capacidad de coordinar el resto de elementos del sistema (*Commander*). Al iniciar el sistema debe identificar todos los instrumentos, administrar el *autotest* y la secuencia de inicialización del sistema, configurar el mapa de direcciones, establecer las jerarquías *Commander/Servant* y empezar la ejecución normal de él mismo.
- El *sub-bus* de alimentación proporciona la potencia necesaria para el funcionamiento del instrumento (hasta 268W a una única tarjeta), en siete tensiones reguladas (+24V, -24V, -5.2V, -2V, además de las normalizadas de *VME*, +12V, -12V y +5V).

La estructura básica de un sistema *VXI* es el chasis. Contiene 13 ranuras en las cuales varios módulos (instrumentos) pueden ser introducidos. El chasis también cubre todas las necesidades de alimentación para el *rack* y los instrumentos que contiene. Los instrumentos *VXI* encajan perfectamente en las ranuras del chasis.

En la norma *VXI* no se determina únicamente el protocolo de comunicaciones y la conexión física de los instrumentos, también son normalizados aspectos eléctricos, mecánicos, de compatibilidad electromagnética y tipos de instrumentos. Esta norma está basada en el *bus VME*, en su velocidad, flexibilidad, estructura jerárquica y la capacidad integración de un conjunto heterogéneo de tarjetas. Como consecuencia directa de apoyar su diseño en el *VME*, los elementos o sistemas *VME* se pueden utilizar en las aplicaciones *VXI*.

La norma facilita múltiples configuraciones posibles. Las más comunes son: un único controlador interno que realiza las funciones de *Slot0* y *Resource Manager*; un único controlador externo (*Host*) conectado en la ranura 0 y el *Resource Manager* conexionado a través de una interfaz *IEEE-488*, local o *RS232*; un conjunto de procesadores internos con control distribuido de sistemas.

Se han declarado cuatro tipos de instrumentos *VXI*: basados en registros, basados en mensajes, de memoria y extendidos. Cada uno de ellos puede estar formado por varios módulos y en términos de *software* es necesario coordinarlos adecuadamente y compartir su información de manera transparente, natural y sencilla.

Los dispositivos basados en registros se comunican con su módulo controlador mediante protocolos especificados por el fabricante escribiendo en sus registros de comunicación o configuración.

Los elementos basados en mensajes, se interrelacionan mediante mensajes de caracteres *ASCII* usando el protocolo *Word Serial Protocol (WSP)*.

Los instrumentos de memoria ofrecen espacio de almacenamiento de datos en bloques de memoria *ROM* o *RAM*.

Este *bus* está diseñado para aplicaciones de un alto grado de integración y una alta velocidad, que necesitaba mucho más que el estándar de comunicación *IEEE-488*; sin embargo, dada su popularidad, se inspiró en él.

El tercer *bus* estándar de la industria fue *PXI*. Introducido en 1997 por *National Instruments* y adoptado como estándar en 1998, *PXI* está diseñado para conseguir un gran desempeño en la medida y la automatización de aplicaciones que requieren un factor de forma resistente.

El estándar más novedoso*, LXI*, fue lanzado por el consorcio de *LXI* en 2005. *LXI* es el sucesor basado en redes *LAN* del *GPIB* y añade más características a instrumentos *LAN*. Construyendo sistemas basados en estándares abiertos de la industria, los ingenieros pueden elegir los componentes para sus sistemas basados en sus necesidades independientemente del vendedor. Los estándares abiertos también permites el uso de sistemas híbridos. Los ingenieros pueden usar *VXI* para una parte de su aplicación y *GPIB* o *PXI* para otra parte gracias a la interoperabilidad de los estándares abiertos.

A.1.2 Especificaciones *VXI*

A.1.2.1 Especificaciones mecánicas

La especificación del *VXIbus* determina las dimensiones deben poseer los módulos de *VXI* para compatibilizarlo con la estructura del *backplane*. Hay cuatro tamaños distintos [22] [55]:

- Tamaño A: 10 x 16 cm con un espaciado de ranura de 2 cm.
- Tamaño B: 23,3 x 16 cm con un espaciado de ranura de 3 cm.
- Tamaño C: 23,3 x 34 cm con un espaciado de ranura de 3 cm.
- Tamaño D: 36,7 x 34 cm con un espaciado de ranura de 3 cm.

Los dos primeros tamaños (A y B), de menor envergadura, son compatibles con el *VMEbus* y no gozan de todos los beneficios del estándar de *VXI*, pero ofrecen una buena compensación en coste y complejidad, lo que les convierte en tarjetas útiles para instrumentos simples y baratos. Las otras tarjetas de mayor magnitud (C y D) posibilitan un mayor rendimiento de la instrumentación, así como un mayor aislamiento de circuitos sensibles y un aumento de la precisión.

El tamaño C se ha convertido en el más utilizado, alrededor del 90% de los productos *VXI* son de estas dimensiones, ya que permite un gran desempeño y resulta una tarjeta más compacta. Sin embargo, el tamaño D también se utiliza, en instrumentos de pruebas funcionales de grandes dimensiones, a pesar de su alto coste.

A.1.2.2 Especificaciones eléctricas

VMEbus marca las especificaciones eléctricas de *VXI*; sin embargo, se han añadido varios voltajes de alimentación, circuitos *ECL* (*Emitter-Coupled Logic*), *bus*es de instrumentación para sincronización de medidas y disparo, junto con un *bus* adicional y un conjunto de líneas de *bus* locales para la comunicación módulo-módulo (privada).

El *VXIbus* tiene tres conectores *DIN* (antiguamente de 96 pines y actualmente de 160 pines, para introducir más fuentes de alimentación, tierras y señales para nuevos protocolos de transferencia de bloques en las filas de los extremos, protegiendo la compatibilidad), conservando la disposición de los pines pertenecientes al *VMEbus*, tal y como muestran las tablas 12, 13, 14, 15 y 16 [55]:

- El conector P1 es de existencia obligatoria, es el soporte para el *bus* transferencia de datos (hasta 24 bits de direccionamiento y 16 bits de datos), el *bus* de interrupción y parte de la potencia. Está completamente determinado por las especificaciones del *VMEbus*.
- El conector P2 es opcional, disponible en tarjetas de superior envergadura al tamaño A. Amplia el *bus* de trasferencia de datos a 32 bits y agrega la estructura necesaria para cuatro voltajes de alimentación, el *bus* local, el *bus* de identificación de módulo, el *bus* analógico adicional (que recorre el *backplane*), *bus*es de disparo *TTL* y *ECL* (que

se sitúan a lo largo del *backplane* y están definidos por cuatro protocolos) y una señal de reloj *ECL* diferencial de 10 MHz (que es llevada a cada *slot*).

- El conector P3 es opcional, únicamente disponible en el tamaño D, expande los recursos del conector P2 para aplicaciones especializadas permitiendo un mayor desempeño de la instrumentación. Proporciona 24 líneas de *bus* local adicionales (48 pines), líneas de disparo *ECL* complementarias, una señal de reloj de 100MHz y líneas de disparo para una sincronización de precisión.

Las líneas de disparo se usan principalmente como recurso para las señales entre instrumentos, mientras que las líneas de *bus* local se utilizan preferentemente dentro de un conjunto de instrumentos múltiples en módulos.

El *bus* local es un *bus* conectado en serie que añade capacidades significantes a los sistemas de medida, con una amplitud de 12 líneas a cada dirección a través del conector P2 y 24 líneas adicionales a través del conector P3. Permite señales *TTL*, *ECL*, analógicas desde baja tensión a 42 voltios. Una muesca cerca de la platina previene la inserción errónea de otra clase de conectores. Internamente, cada *slot* del ordenador central *VXI* o mainframe, tiene un conjunto de líneas de transmisión de 50 ohmios entre las ranuras adyacentes a cada lado. Las tablas 12, 13, 14, 15 y 16 aclaran estos detalles.

PIN NUMBER	ROW 'z' SIGNAL MNEMONIC	ROW 'a' SIGNAL MNEMONIC	ROW 'b' SIGNAL MNEMONIC	ROW 'c' SIGNAL MNEMONIC	ROW 'd' SIGNAL MNEMONIC
1	MPR	D00	BBSY*	D08	VPC (1)
2	GND	D01	BCLR*	D09	GND (1)
3	MCLK	D02	ACFAIL*	D10	+V1
4	GND	D03	BG0IN*	D11	+V2
5	MSD	D04	BG0OUT*	D12	RsvU
6	GND	D05	BG1IN*	D13	-V1
7	MMD	D06	BG1OUT*	D14	-V2
8	GND	D07	BG2IN*	D15	RsvU
9	MCTL	GND	BG2OUT*	GND	GAP*
10	GND	SYSCLK	BG3IN*	SYSFAIL*	GA0*
11	RESP*	GND	BG3OUT*	BERR*	GA1*
12	GND	DS1*	BR0*	SYSRESET*	+3.3V
13	RsrvBus	DS0*	BR1*	LWORD*	GA2*
14	GND	WRITE*	BR2*	AM5	+3.3V
15	RsrvBus	GND	BR3*	A23	GA3*
16	GND	DTACK*	AM0	A22	+3.3V
17	RsrvBus	GND	AM1	A21	GA4*
18	GND	AS*	AM2	A20	+3.3V
19	RsrvBus	GND	AM3	A19	RsrvBus
20	GND	IACK*	GND	A18	+3.3V
21	RsrvBus	IACKIN*	SERA	A17	RsrvBus
22	GND	IACKOUT*	SERB	A16	+3.3V
23	RsrvBus	AM4	GND	A15	RsrvBus
24	GND	A07	IRQ7*	A14	+3.3V
25	RsrvBus	A06	IRQ6*	A13	RsrvBus
26	GND	A05	IRQ5*	A12	+3.3V
27	RsrvBus	A04	IRQ4*	A11	LI/I*
28	GND	A03	IRQ3*	A10	+3.3V
29	RsrvBus	A02	IRQ2*	A09	LI/O*
30	GND	A01	IRQ1*	A08	+3.3V
31	RsrvBus	-12V	+5V STDBY	+12V	GND (1)
32	GND	+5V	+5V	+5V	VPC (1)

Tabla 12. Señales asignadas en el conector *P1* de *VXIbus* [55]

PIN NUMBER	ROW 'z' SIGNAL MNEMONIC	ROW 'a' SIGNAL MNEMONIC	ROW 'b' SIGNAL MNEMONIC	ROW 'c' SIGNAL MNEMONIC	ROW 'd' SIGNAL MNEMONIC
1	+V3	ECLTRG0	+5V	CLK10+	GND (1)
2	GND	-2V	GND	CLK10-	GND (1)
3	-V3	ECLTRG1	RSV1	GND	LCLK100+
4	GND	GND	A24	-5.2V	LCLK100-
5	+V4	MODID12	A25	LBUSC00	GND
6	GND	MODID11	A26	LBUSC01	LSYNC100+
7	-V4	-5.2V	A27	GND	LSYNC100-
8	GND	MODID10	A28	LBUSC02	GND
9	SDA0	MODID09	A29	LBUSC03	STRGIN12+
10	GND	GND	A30	GND	STRGIN12-
11	SCL0	MODID08	A31	LBUSC04	STRGIN11+
12	GND	MODID07	GND	LBUSC05	STRGIN11-
13	SDA1	-5.2V	+5V	-2V	STRGIN10+
14	GND	MODID06	D16	LBUSC06	STRGIN10-
15	SCL1	MODID05	D17	LBUSC07	STRGIN09+
16	GND	GND	D18	GND	STRGIN09-
17	STRGIN04+	MODID04	D19	LBUSC08	STRGIN08+
18	GND	MODID03	D20	LBUSC09	STRGIN08-
19	STRGIN04-	-5.2V	D21	-5.2V	STRGIN07+
20	GND	MODID02	D22	LBUSC10	STRGIN07-
21	STRGIN03+	MODID01	D23	LBUSC11	STRGIN06+
22	GND	GND	GND	GND	STRGIN06-
23	STRGIN03-	TTLTRG0*	D24	TTLTRG1*	STRGIN05+
24	GND	TTLTRG2*	D25	TTLTRG3*	STRGIN05-
25	STRGIN02+	+5V	D26	GND	STRGOUT0+
26	GND	TTLTRG4*	D27	TTLTRG5*	STRGOUT0-
27	STRGIN02-	TTLTRG6*	D28	TTLTRG7*	STRGOUT1+
28	GND	GND	D29	GND	STRGOUT1-
29	STRGIN01+	RSV2	D30	RSV3	STRGOUT2+
30	GND	MODID00	D31	GND	STRGOUT2-
31	STRGIN01-	GND	GND	+24V	GND (1)
32	GND	SUMBUS	+5V	-24V	VPC (1)

Tabla 13. Señales asignadas en el conector *P2* de *VXIbus* en el *slot* 0 [55]

PIN NUMBER	ROW 'z' SIGNAL MNEMONIC	ROW 'a' SIGNAL MNEMONIC	ROW 'b' SIGNAL MNEMONIC	ROW 'c' SIGNAL MNEMONIC	ROW 'd' SIGNAL MNEMONIC
1	+V3	ECLTRG0	+5V	CLK10+	GND(1)
2	GND	-2V	GND	CLK10-	GND(1)
3	-V3	ECLTRG1	RSV1	GND	LCLK100+
4	GND	GND	A24	-5.2V	LCLK100-
5	+V4	LBUSA00	A25	LBUSC00	GND
6	GND	LBUSA01	A26	LBUSC01	LSYNC100+
7	-V4	-5.2V	A27	GND	LSYNC100-
8	GND	LBUSA02	A28	LBUSC02	GND
9	LBUSA12	LBUSA03	A29	LBUSC03	GND
10	GND	GND	A30	GND	LBUSC12
11	LBUSA13	LBUSA04	A31	LBUSC04	LBUSC13
12	GND	LBUSA05	GND	LBUSC05	+5V
13	LBUSA14	-5.2V	+5V	-2V	LBUSC14
14	GND	LBUSA06	D16	LBUSC06	LBUSC15
15	LBUSA15	LBUSA07	D17	LBUSC07	GND
16	GND	GND	D18	GND	LBUSC16
17	LBUSA16	LBUSA08	D19	LBUSC08	LBUSC17
18	GND	LBUSA09	D20	LBUSC09	+5V
19	LBUSA17	-5.2V	D21	-5.2V	LBUSC18
20	GND	LBUSA10	D22	LBUSC10	LBUSC19
21	LBUSA18	LBUSA11	D23	LBUSC11	GND
22	GND	GND	GND	GND	GND
23	LBUSA19	TTLTRG0*	D24	TTLTRG1*	STRGOUT+
24	GND	TTLTRG2*	D25	TTLTRG3*	STRGOUT-
25	+12V	+5V	D26	GND	STRGIN0+
26	GND	TTLTRG4*	D27	TTLTRG5*	STRGIN0-
27	-12V	TTLTRG6*	D28	TTLTRG7*	STRGIN1+
28	GND	GND	D29	GND	STRGIN1-
29	+24V	RSV2	D30	RSV3	STRGIN2+
30	GND	MODID	D31	GND	STRGIN2-
31	-24V	GND	GND	+24V	GND (1)
32	GND	SUMBUS	+5V	-24V	VPC (1)

Tabla 14. Señales asignadas en el conector *P2* de *VXIbus* en los *slots* 1-12 [55]

PIN NUMBER	ROW 'z' SIGNAL MNEMONIC	ROW 'a' SIGNAL MNEMONIC	ROW 'b' SIGNAL MNEMONIC	ROW 'c' SIGNAL MNEMONIC	ROW 'd' SIGNAL MNEMONIC
1	RsrvBus	ECLTRG2	+24V	+12V	RsrvBus
2	GND	GND	-24V	-12V	RsrvBus
3	RsrvBus	ECLTRG3	GND	RSV4	RsrvBus
4	GND	-2V	RSV5	+5V	RsrvBus
5	RsrvBus	ECLTRG4	-5.2V	RSV6	RsrvBus
6	GND	GND	RSV7	GND	RsrvBus
7	RsrvBus	ECLTRG5	+5V	-5.2V	RsrvBus
8	GND	-2V	GND	GND	RsrvBus
9	RsrvBus	STARX12+	+5V	STARX01+	RsrvBus
10	GND	STARX12-	STARY01-	STARX01-	RsrvBus
11	RsrvBus	STARY12+	STARY12-	STARY01+	RsrvBus
12	GND	STARX11+	GND	STARX02+	RsrvBus
13	RsrvBus	STARX11-	STARY02-	STARX02-	RsrvBus
14	GND	STARY11+	STARY11-	STARY02+	RsrvBus
15	RsrvBus	STARX10+	+5V	STARX03+	RsrvBus
16	GND	STARX10-	STARY03-	STARX03-	RsrvBus
17	RsrvBus	STARY10+	STARY10-	STARY03+	RsrvBus
18	GND	STARX09+	-2V	STARX04+	RsrvBus
19	RsrvBus	STARX09-	STARY04-	STARX04-	RsrvBus
20	GND	STARY09+	STARY09-	STARY04+	RsrvBus
21	RsrvBus	STARX08+	GND	STARX05+	RsrvBus
22	GND	STARX08-	STARY05-	STARX05-	RsrvBus
23	RsrvBus	STARY08+	STARY08-	STARY05+	RsrvBus
24	GND	STARX07+	+5V	STARX06+	RsrvBus
25	RsrvBus	STARX07-	STARY06-	STARX06-	RsrvBus
26	GND	STARY07+	STARY07-	STARY06+	RsrvBus
27	RsrvBus	GND	GND	GND	RsrvBus
28	GND	STARY+	-5.2V	STARX+	RsrvBus
29	RsrvBus	STARY-	GND	STARX-	RsrvBus
30	GND	GND	-5.2V	-5.2V	RsrvBus
31	RsrvBus	CLK100+	-2V	SYNC100+	RsrvBus
32	GND	CLK100-	GND	SYNC100-	RsrvBus

Tabla 15. Señales asignadas en el conector *P3* de *VXIbus* en el *slot* 0 [55]

PIN NUMBER	ROW 'z' SIGNAL MNEMONIC	ROW 'a' SIGNAL MNEMONIC	ROW 'b' SIGNAL MNEMONIC	ROW 'c' SIGNAL MNEMONIC	ROW 'd' SIGNAL MNEMONIC
1	RsrvBus	ECLTRG2	+24V	+12V	RsrvBus
2	GND	GND	-24V	-12V	RsrvBus
3	RsrvBus	ECLTRG3	GND	RSV4	RsrvBus
4	GND	-2V	RSV5	+5V	RsrvBus
5	RsrvBus	ECLTRG4	-5.2V	RSV6	RsrvBus
6	GND	GND	RSV7	GND	RsrvBus
7	RsrvBus	ECLTRG5	+5V	-5.2V	RsrvBus
8	GND	-2V	GND	GND	RsrvBus
9	RsrvBus	LBUSA12	+5V	LBUSC12	RsrvBus
10	GND	LBUSA13	LBUSC15	LBUSC13	RsrvBus
11	RsrvBus	LBUSA14	LBUSA15	LBUSC14	RsrvBus
12	GND	LBUSA16	GND	LBUSC16	RsrvBus
13	RsrvBus	LBUSA17	LBUSC19	LBUSC17	RsrvBus
14	GND	LBUSA18	LBUSA19	LBUSC18	RsrvBus
15	RsrvBus	LBUSA20	+5V	LBUSC20	RsrvBus
16	GND	LBUSA21	LBUSC23	LBUSC21	RsrvBus
17	RsrvBus	LBUSA22	LBUSA23	LBUSC22	RsrvBus
18	GND	LBUSA24	-2V	LBUSC24	RsrvBus
19	RsrvBus	LBUSA25	LBUSC27	LBUSC25	RsrvBus
20	GND	LBUSA26	LBUSA27	LBUSC26	RsrvBus
21	RsrvBus	LBUSA28	GND	LBUSC28	RsrvBus
22	GND	LBUSA29	LBUSC31	LBUSC29	RsrvBus
23	RsrvBus	LBUSA30	LBUSA31	LBUSC30	RsrvBus
24	GND	LBUSA32	+5V	LBUSC32	RsrvBus
25	RsrvBus	LBUSA33	LBUSC35	LBUSC33	RsrvBus
26	GND	LBUSA34	LBUSA35	LBUSC34	RsrvBus
27	RsrvBus	GND	GND	GND	RsrvBus
28	GND	STARY+	-5.2V	STARY+	RsrvBus
29	RsrvBus	STARY-	GND	STARY+	RsrvBus
30	GND	GND	-5.2V	-5.2V	RsrvBus
31	RsrvBus	CLK100+	-2V	SYNC100+	RsrvBus
32	GND	CLK100-	GND	SYNC100-	RsrvBus

Tabla 16. Señales asignadas en el conector *P3* de *VXIbus* en los *slots* 1-12 [55]

La longevidad del *bus VXI* ha permitido múltiples perfeccionamientos manteniendo la compatibilidad y adoptando nuevas tecnologías que mejoran el rendimiento y la flexibilidad. Cuando las transformaciones evolutivas no son capaces de dar abasto a las demandas sobre los sistemas de test, la interconexión entre módulos necesita ser modernizada. Por ello se ha añadido un conector adicional (P0), que pretende adoptar una interconexión de plano de datos usando tecnologías serie conmutadas, con los consecuentes beneficios [55]:

- Mayor ancho de banda de operación.
- Aumento del ancho de banda agregado.
- Reducción de la latencia de enlace.

- Menor disputa por el medio de interconexión.
- Incremento de la expansibilidad.
- Disminución del enrutamiento consumido.

El conector *P0* se ajusta al estándar serie conmutado de *VME* (*VXS*), que precisa las características físicas que posibilitan una alta velocidad en el *VXIbus*, situándose entre el conector *P1* y *P2*.

Como se puede observar en las tablas 17 y 18 las filas F y G del conector del *backplane*, se combinan para formar la fila E del conector del módulo, al igual que las filas B Y C del *backplane* se unen para constituir la fila B del conector del módulo. Esto ofrece una toma doble de tierra y una mejor integridad de la señal.

	Row G	Row F	Row E	Row D	Row C	Row B	Row A
1	PA_SCL	GND	PA_TX0-	PA_TX0+	GND	PA_RX0-	PA_RX0+
2	GND	PA_TX1-	PA_TX1+	GND	PA_RX1-	PA_RX1+	GND
3	PA_SDA	GND	PA_TX2-	PA_TX2+	GND	PA_RX2-	PA_RX2+
4	GND	PA_TX3-	PA_TX3+	GND	PA_RX3-	PA_RX3+	GND
5	RFU	GND	PA_SGTX-	PA_SGTX+	GND	PA_SGRX-	PA_SGRX+
6	GND	RFU	RFU	GND	RFU	RFU	GND
7	RFU	GND	RFU	RFU	GND	RFU	RFU
8	GND	RFU	RFU	GND	RFU	RFU	GND
9	RFU	GND	RFU	RFU	GND	RFU	RFU
10	GND	RFU	RFU	GND	RFU	RFU	GND
11	PEN*	GND	PB_SGTX-	PB_SGTX+	GND	PB_SGRX-	PB_SGRX+
12	GND	PB_TX0-	PB_TX0+	GND	PB_RX0-	PB_RX0+	GND
13	PB_SCL	GND	PB_TX1-	PB_TX1+	GND	PB_RX1-	PB_RX1+
14	GND	PB_TX2-	PB_TX2+	GND	PB_RX2-	PB_RX2+	GND
15	PB_SDA	GND	PB_TX3-	PB_TX3+	GND	PB_RX3-	PB_RX3+

Tabla 17. Señales asignadas en el módulo [55]

	Row A	Row B	Row C	Row D	Row E	Row F	Row G	Row H	Row I
1	PA_RX0+	PA_RX0-	GND	GND	PA_TX0+	PA_TX0-	GND	GND	PA SCL
2	GND	GND	PA_RX1+	PA_RX1-	GND	GND	PA_TX1+	PA_TX1-	GND
3	PA_RX2+	PA_RX2-	GND	GND	PA_TX2+	PA_TX2-	GND	GND	PA SDA
4	GND	GND	PA_RX3+	PA_RX3-	GND	GND	PA_TX3+	PA_TX3-	GND
5	PA_SGRX+	PA_SGRX-	GND	GND	PA_SGTX+	PA_SGTX-	GND	GND	RFU
6	GND	GND	RFU	RFU	GND	GND	RFU	RFU	GND
7	RFU	RFU	GND	GND	RFU	RFU	GND	GND	RFU
8	GND	GND	RFU	RFU	GND	GND	RFU	RFU	GND
9	RFU	RFU	GND	GND	RFU	RFU	GND	GND	RFU
10	GND	GND	RFU	RFU	GND	GND	RFU	RFU	GND
11	PB_SGRX+	PB_SGRX-	GND	GND	PB_SGTX+	PB_SGTX-	GND	GND	PEN*
12	GND	GND	PB_RX0+	PB_RX0-	GND	GND	PB_TX0+	PB_TX0-	GND
13	PB_RX1+	PB_RX1-	GND	GND	PB_TX1+	PB_TX1-	GND	GND	PB SCL
14	GND	GND	PB_RX2+	PB_RX2-	GND	GND	PB_TX2+	PB_TX2-	GND
15	PB_RX3+	PB_RX3-	GND	GND	PB_TX3+	PB_TX3-	GND	GND	PB SDA

Tabla 18. Señales asignadas en el *backplane* [55]

A.1.2.3 Especificaciones electromagnéticas

La especificación *VXIbus* estipula unos límites estrictos de compatibilidad electromagnética (*EMC*) tanto conducidos como irradiados, establecen la cantidad de interferencia que un dispositivo puede emitir. Estos límites aseguran que los módulos electrónicamente sensibles operen de acuerdo a la previsión sin interferencias provenientes de otro módulo que se encuentre en el sistema. [55]

Por ello, los módulos y mainframes deben precisar sus emisiones a alta frecuencia y su susceptibilidad.

A.1.2.4 Especificaciones de alimentación y refrigeración

Para asegurar una refrigeración adecuada en un sistema *VXI*, cada fabricante de mainframe elabora una documentación que presenta un gráfico de refrigeración para el peor caso de la configuración del módulo, que especifica en términos de presión en el módulo en comparación con el flujo de aire creado, el flujo de aire y la presión necesaria para que el instrumento funcione adecuadamente. A través de este gráfico, se localiza el punto que representa las condiciones del dispositivo, y si se encuentra entre los límites, garantiza su compatibilidad.

Las especificaciones de alimentación facilitan la integración. Cada mainframe e instrumento *VXI* detalla en qué rango debe ser alimentado y la corriente estacionaria y dinámica que proporciona a diferentes voltajes. Cada alimentación muestra un pico de suministro (de corriente alterna y corriente continua respectivamente) tanto de corriente como de tensión, que al ser comparados entre ellos (la capacidad del mainframe *VXI* con la capacidad de los módulos) determinan la compatibilidad. Además, la especificación de corriente dinámica precisa que los módulos no induzcan más ruido ondulatorio en las alimentaciones del mainframe del que ningún otro módulo pueda soportar. [55]

La mayoría de las fuentes de alimentación del mainframe se ajustan a la siguiente tabla:

VOLTAGE	DESCRIPTION	ALLOWED VARIATION	DC LOAD RIPPLE/NOISE	INDUCED RIPPLE/NOISE
+5 V	+5 VDC	+0.25 V/-0.125 V	50 mV	50 mV
+3.3V	+3.3V	+0.3 V/-0.3 V	50 mV	50 mV
+12 V	+12 VDC	+0.60 V/-0.36 V	50 mV	50 mV
-12 V	-12 VDC	-0.60 V/+0.36 V	50 mV	50 mV
+24 V	+24 VDC	+1.20 V/-0.72 V	150 mV	150 mV
-24 V	-24 VDC	-1.20 V/+0.72 V	150 mV	150 mV
-5.2 V	-5.2 VDC	-0.260 V/-0.156 V	50 mV	50 mV
-2 V	-2 VDC	-0.10 V/+0.10 V	50 mV	50 mV
+5 V STDBY	+5 VDC standby	+0.25 V/-0.125 V	50 mV	50 mV

Tabla 19. Especificaciones de tensión de alimentación [55]

La alimentación debe ser distribuida en el *backplane* en voltajes de corriente continua:

- *+5 VDC*, usado especialmente para comunicaciones de *bus* y como principal suministro de energía (utilizando posteriormente un regulador a *+3,3 VDC* o sin él).
- *+3,3 VDC*, capaz de sustituir la regulación de *+5 VDC*.
- *±12 VDC*, utilizado principalmente por dispositivos analógicos, unidades de disco e interfaces de comunicación.
- *±24 VDC*, empleado normalmente para alimentar fuentes de señales analógicas y conseguir otros niveles de tensión.
- *-5,2 VDC*, dedicado a los dispositivos *ECL*.
- *-2 VDC*, aprovechado para la terminación de las cargas *ECL*.
- *+5 VDC STDBY*, destinado a sustentar la memoria, relojes, etc. Cuando no se encuentran los *+5 VDC*.

A.1.2.5 Especificaciones de comunicaciones

VXIbus especifica diversos tipos de dispositivos y protocolos, así como el acuerdo de comunicación, con una gran flexibilidad inherente a una arquitectura abierta. Un sistema *VXI* puede ser controlado usando un ordenador embebido o un ordenador externo compatible con la plataforma, es este último caso la interfaz del mainframe puede ser flexible (*GPIB*/*VXI*, *MXI*/*VXI*, *RS-232*/*VXI*, *Ethernet*/*VXI*, etc.) con sus correspondientes ventajas e inconvenientes.

Aunque los instrumentos conectados en el mainframe pueden ser identificados por el número de ranura en la que están insertados, existe una dirección lógica única (*ULA*) para cada dispositivo *VXI* (con un rango de 0 a 255) que permite reconocer el instrumento y no tiene ninguna relación con el *slot* en el que esté introducido. Esto posibilita crear un sistema con hasta 256 dispositivos con mayor portabilidad e integración. [25]

Los módulos *VXI* albergan estos datos en los registros localizados en las direcciones específicas, los primeros 16kB de los 64kB totales son reservados para la configuración de los dispositivos *VXI*, 8 bits son destinados a determinar la dirección lógica de cada instrumento y el resto es dependiente del tipo de instrumento.

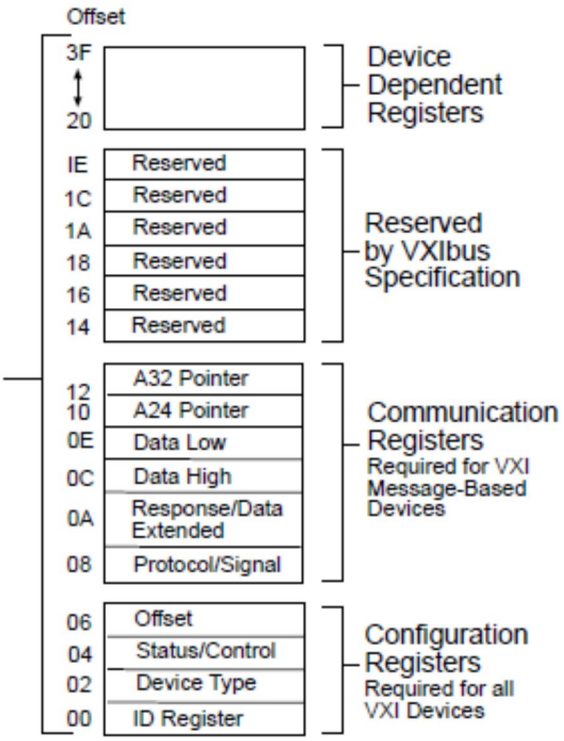


Tabla 20. Configuración del espacio de los registros en *VXI* [25]

Fundamentalmente existen cuatro tipos de instrumento [55]:

- Dispositivo basado en registros. Es el dispositivo sencillo y se suele utilizar como base para módulos simples. Se comunica solo a través de lecturas y escrituras de registros y su configuración es manejada por elementos del *VXIbus*, pero programado mediante registros dependientes del dispositivo. Esto significa que se programan a un nivel bajo usando información binaria; sin embargo, *VXIbus* facilita una herramienta (*Commanders and Servants*) que permite identificar como *Commander* a un dispositivo capaz de manejar a otro basado en registros. Así el *Commander* interpreta la información a un nivel alto e inmediatamente se la comunica al instrumento basado en registros.
  El sistema puede identificar cada dispositivo, su tipo, modelo y fabricante, dirección, y requisitos de memoria.
- Dispositivo basado en mensajes. Es un instrumento más inteligente y más caro. Además de la configuración básica de registros, posee elementos de comunicación comunes y un protocolo serie para permitir el intercambio de información con caracteres *ASCII* con otros módulos basados en mensajes.
  El sacrificio de la velocidad para interpretar el *ASCII* facilita inmensamente el soporte, por lo que los dispositivos basados en mensajes están normalmente limitados a velocidades de la norma *IEEE-488*.
  El protocolo serie de palabra es un protocolo estandarizado de paso de mensajes muy similar al protocolo *IEEE-488*. Envía solo un byte por transición, que debe ser interpretado generalmente por un microprocesador, por su parecido al transmisor receptor asíncrono universal (*UART*) en un puerto serie.

La complejidad de estos dispositivos les permite realizar tareas enrevesadas y establecer canales de comunicación (como canales de memoria compartidos), para aprovechar al máximo las capacidades del *VXIbus*.

- Dispositivo de memoria. Es un instrumento basado en registros especializado y optimizado para mantener y mover grandes cantidades de datos.
- Dispositivos extendidos. Este término constituye un tipo de instrumento reservado y proporciona un camino de desarrollo a nuevos tipos de dispositivos en el futuro.

Aunque las diferencias entre unos dispositivos y otros aparentan ser descomunales, todos comparten una base común, sobre la que se han añadido las funcionalidades que los convierten en unos tipos diferentes de instrumentos (véase la figura 49).

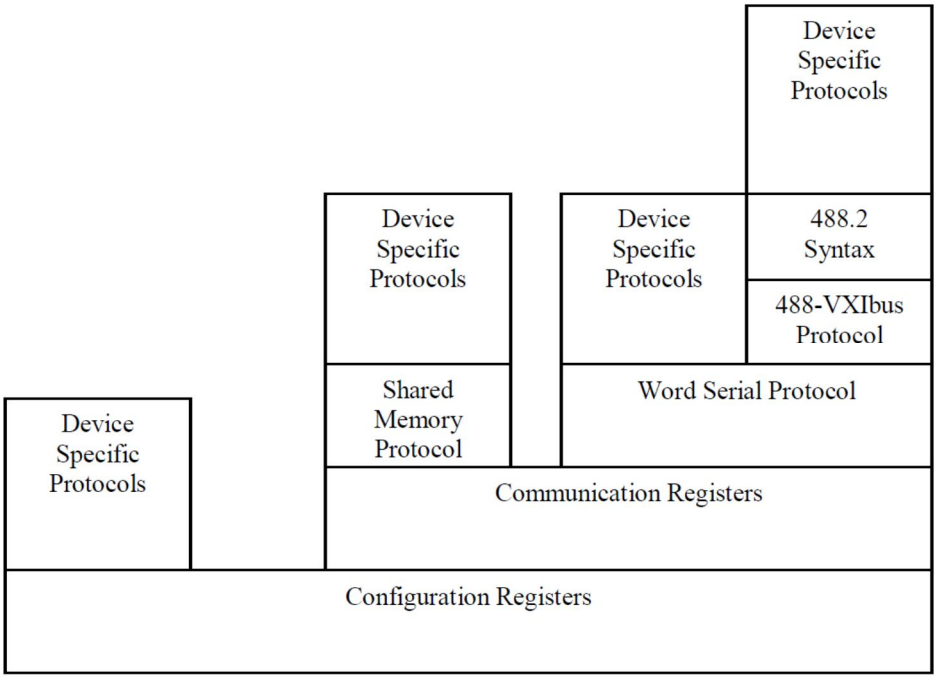


Figura 49. Capas de comunicación de *VXIbus* [25]

A.2 Arquitecturas de comunicación

Las arquitecturas de comunicación son estructuras organizadas jerárquicamente, con el propósito de permitir el intercambio de información entre niveles lógicos similares en diferentes máquinas o terminales de una red. En una arquitectura basada en capas es necesario un mecanismo para posibilitar la relación entre capas. Una interfaz proporciona los métodos de acceso a los distintos servicios que cada capa provee, de manera que mediante un sistema de llamadas y respuestas, se interconectan las capas.

Dada la diversidad de tecnologías y fabricantes es imprescindible contar con estandarizaciones que permitan la interconectividad. Los estándares son fundamentales para construir y sustentar un mercado competitivo y abierto entre los distintos fabricantes de equipos, y para asegurar la interoperabilidad de la información, los procesos de comunicación y la tecnología.

A.2.1 Estándares de comunicación serie

La comunicación serie consiste en el envío de un bit de información de manera secuencial entre dispositivos, es decir, de un bit en un bit a una velocidad pactada entre el emisor y el receptor. Es un método de comunicación sencillo y puede alcanzar una distancia de hasta 1200 metros, tal y como establece la especificación *IEEE 488*.

Normalmente se suele utilizar para la transferencia de datos en formato *ASCII*, para lo que usa tres líneas: Referencia (Tierra o *GND*), Transmitir (*TX*) y Recibir (*RX*). El control de flujo puede llevarse a cabo a nivel de *software*, como el uso de *XON-XOFF* (generalmente los caracteres *ASCII* 17 y 19 respectivamente), o a nivel de *hardware* con líneas de control adicionales. En la comunicación serie es habitual una transmisión asíncrona y es posible transmitir y recibir a la vez, aunque existen otras líneas utilizables para *handshaking* (intercambio de pulsos de sincronización).

Las principales características que marcan la comunicación serie son [23]:

- Bits de datos. Es la cantidad de bits transferidos, por lo general suelen ser 8 bits (necesarios para el estándar *ASCII* extendido); sin embargo, también se usan bastante 5 bits y 7 bits (para el estándar *ASCII*). Estos bits se agrupan en paquetes incorporando bits de inicio/parada y paridad.
- Bits de inicio/parada. Aprovechados para indicar el comienzo/fin de la transferencia de un paquete y proporcionar un margen de error en casi de que los relojes de los dispositivos no estén sincronizados. Frecuentan los tamaños (en orden creciente de tolerancia) de 1bit, 1.5 bits o 2 bits.
- Paridad. Se trata de una verificación opcional de errores para asegurar la comunicación frente a ruidos o desincronizaciones, clasificable en cuatro tipos:
    - Par, en la que el puerto fija el bit de paridad, o sea, el último bit tras los bits de datos, de forma que el número de bits en estado lógico alto sea par.

  - Impar, en la que el puerto asigna el bit de paridad, de manera que el número de bits en estado lógico alto sea impar.
  - Marcada, en la que el puerto determina el bit de paridad en estado lógico alto.
  - Espaciada, en la que el puerto precisa el bit de paridad en estado lógico bajo.
- Velocidad de transmisión (*Baud rate*). Especifica el número de bits por segundo transmitidos, en baudios ($^{Bits}/_{segundo}$). Las velocidades más comunes son de 14400, 28800 y 33600 baudios; sin embargo, cuando los dispositivos se encuentran situados muy cerca uno del otro, entonces es posible utilizar velocidades más elevadas.

Dada la popularidad de la comunicación serie, gracias a las actuales velocidades de transferencia, la buena integridad de la señal y la sencillez frente a otro tipo de comunicaciones, existen múltiples protocolos estandarizados, entre los que destacan:

- *RS-232*, es el estándar más utilizado gracias a que cada ordenador está equipado con al menos un puerto serie y las señales comparten una tierra común. Solo es posible realizar conexiones punto a punto, es decir, un emisor y un receptor.
- *RS-422*, es un interfaz de alta velocidad digital con una configuración de cableado diferencial y en pares trenzados, que permite que cualquier ruido o interferencia en un cable sea transmitido al otro, lo que posibilita su anulación.
- *RS-485*, es la versión del *RS-232* más mejorada disponible. Usa señales diferenciales a través de cables de par trenzado, que minimiza los errores producidos por el ruido o las interferencias. Es capaz de comunicarse a altas velocidades, a través de cables a gran distancia, y con tecnología multipunto, que permite conectar hasta 32 dispositivos.

A.2.1.1 Estándar de comunicaciones RS-232

A.2.1.1.1 Fundamentos

El *Recommended Standard 232* desarrollado por *Electronics Industries Association* (*EIA*) está definido en las especificaciones ANSI como el interfaz entre un equipo terminal de datos y un equipo de comunicación de datos empleando un intercambio en modo serie de datos binarios. En ellas se detallan las normas y reglas necesarias para comunicar en serie dos dispositivos entre sí.

El interfaz serie *RS-232* fue desarrollado para conectar un ordenador a periféricos comunes, como retroproyectores, sensores y actuadores de aplicaciones de automatización industrial y módems. Sin embargo, las aplicaciones del *RS-232* se han extendido más allá, por lo que se han desarrollado limitaciones en el estándar. Existen diversas variaciones que han surgido a lo largo del tiempo en el estándar, que incluyen compatibilidades con cada vez más dispositivos, pero oficialmente la última revisión del estándar es *RS-232*C, que determina: el tipo de conector, las características eléctricas, las longitudes máximas, los niveles de tensión, el protocolo de comunicación y las señales que intervienen en el proceso.

La señal RS-232 guarda relación con la UART (Universal Asynchronous Receiver Transmitter), la señal TTL (Transistor to Transistor Logic) y el transmisor-receptor de línea:

- La *UART* es la intermediaria entre la *CPU* y el exterior. En caso de que no pueda adaptar las tensiones (*TTL)* al estándar se usa un *driver* de línea transmisor-receptor.
- El transmisor-receptor *RS-232* convierte la señal de tensión a un rango desde +3V a +15V, para un nivel lógico bajo o 0 lógico, y desde -3V a -15V, para un nivel lógico alto o 1 lógico.

El control de flujo en las comunicaciones *RS-232* consiste en que un dispositivo, desde un extremo, envía al otro señales para empezar o parar la transmisión, tanto cuando el emisor necesita parar unos instantes, como cuando el *buffer* del receptor se llena. El control de flujo puede implementarse vía *hardware* o vía *software* [23]:

- El control de flujo por *hardware* generalmente usa las líneas *RTS* (*Request To Send*) y *CTS* (*Clear To Send*) para indicar cuando está preparado el dispositivo para recibir datos. Desde el *RTS* de un aparato se envía un 1 lógico al *CTS* del otro dispositivo, para señalar su disponibilidad para admitir información. En ocasiones se conectan las líneas *RTS* y *CTS* directamente en el mismo aparato, lo que implica que los datos se enviarán tan pronto como se solicite el envío, para prevenir errores se emplean las líneas *DSR (Data Set Ready*) y *DTR* (*Data Terminal Ready*), con una conexión cruzada. *DTR* advierte a *DSR* que puede aceptar información.
- El control de flujo por *software* se encarga de enviar las señales *XON/XOFF*, cuando la unidad está lista para recibir y cuando el dispositivo ya no puede recibir (por un *Buffer* casi lleno) respectivamente, a través de los canales de datos. Aunque el control parece tener solo ventajas, consume ancho de banda y si se pierde la señal *XOFF* o *XON*, algunos datos se perderán por *overrun* del *buffer* o nunca se enviarán datos, respectivamente. Así, normalmente solo se usa el control de flujo por *software* cuando la velocidad de comunicación no es muy alta.

A.2.1.1.2 Características

El último estándar oficial, *RS-232*C, muestra las siguientes características [23]:

- Capacidad de carga máxima de 2500pF, por lo que la distancia de transmisión máxima es de hasta 15 m, en este estándar, en otras variaciones del estándar se sobrepasa.
- Cableado fácil y poco costoso.
- Conexión *Half-duplex*, es decir los datos pueden transmitirse bidireccionalmente, pero no de manera simultánea. Sin embargo, es posible configurarlo para trabajar en *Full-duplex*, o sea, transmitir y recibir a la vez, pero se atendría a otra variación de estándar.
- Velocidad de transferencia de hasta 19,2 Kbps, aunque es posible utilizar cualquier velocidad, como hacen otras variaciones de estándar.
- 1 bit de inicio.

- 1 bit, 1.5 bits (si los bits de datos son 5), o 2 bits de parada.
- Bits de datos con 5, 6, 7 u 8 bits.
- Sin paridad.
- Conexión punto a punto (*point to point*), esto significa que cada canal de datos se emplea únicamente para comunicar dos nodos.
- Modo de operación *Single-ended*. Un cable conduce la señal, mientras que el otro es conectado a tierra.

A.2.1.2 Estándar de comunicaciones RS-485

A.2.1.2.1 Fundamentos

El *Recommended Standard 485* fue desarrollado conjuntamente por la *Electronics Industries Association* y la *Telecommunications Industry Association* (*TIA*) y especifica solo características eléctricas del emisor y el receptor. Es empleado en multitud de aplicaciones, como *SCSI-2, SCSI-3, buses* en cabinas de aviones, *Modbus*, *Profibus*, capa física para *DMX512* y protocolos de automatización industrial, etc.

Utiliza la transmisión diferencial (en un rango desde -7V a 12V) para anular las distorsiones electromagnéticas y de fuentes externas, lo que le permite ser usado para transmitir datos a distancias más lejanas y aislar las señales. La transmisión diferencial consiste en separar la señal en dos cables con voltajes opuestos (A y B), y estas señales son restadas al final de la recepción, logrando una gran integridad de la señal frente a entornos con ruidos e interferencias.

El 1 lógico se representa con una tensión inferior en la línea A (- o *TxD-/RxD-*) con respecto a la B (+ o *TxD+/RxD+*), es decir, un voltaje negativo señala estado *OFF*, marca o 1; contrariamente el 0 lógico o espacio es indicado mediante una tensión superior en el terminal A en relación al terminal B.

RS-485 fue diseñado especialmente para aplicaciones que requerían de una conexión a numerosos dispositivos mediante una sola línea, para ello se incluyen *drivers* (seguidores de señales) *tri-state* (un estado de alta impedancia) que son controlados por una línea programable para asegurar que solo un *driver* actúa como tal en un momento dado.

Para prevenir la reflexión de la señal, se utilizan resistencias terminadoras, que coinciden con la impedancia de los nodos de transmisión y recepción, y cuyo valor es especificado por el fabricante del cableado (comúnmente 120 ohmios). Así mismo, las resistencias en serie de pequeña impedancia a lo largo de las líneas de transmisión aumentan ligeramente la calidad de la señal.

Como precaución ante posibles ruidos e interferencias en la transmisión, se suelen usar resistencias a modo de *pull up* y *pull down* en cada cable de datos, que aseguran a desaparición de falsas comunicaciones cuando ningún dispositivo emplea las líneas de comunicación. También se suele limitar el voltaje en modo común, que puede aparecer en las

entradas de los receptores, mediante la conexión de un tercer cable entre la fuente y el receptor.

A.2.1.2.2 Características

El estándar *RS-485* presenta las siguientes características [23]:

- Longitud máxima de alcance de hasta 1220 metros, a 100 kbps.
- Capacidad de hasta 32 estaciones.
- Alimentación única de +5V.
- Conexión *Half-duplex*, convertible a *Full-duplex* usando cuatro cables.
- Velocidad de transferencia de hasta 10 Mbps, a 12 metros.
- Posibilidad de conexión *Multidrop*, que consiste en un enlace en un solo sentido, en el cual un transmisor se comunica con diversos receptores.
- Configuración opcional en conexión multipunto (*multipoint*), esto significa que cada canal de datos se emplea únicamente para comunicar muchos nodos, concretamente hasta 32.

A.2.2 Estándar de red *Ethernet*

A.2.2.1 Fundamentos

Ethernet es una familia de tecnologías que proporciona especificaciones físicas y de enlaces de datos para controlar el acceso a una red compartida, dominante en el ámbito de las redes de área local, por sus múltiples ventajas sobre otras tecnologías de red local:

- Sencillo de instalar y manejar.
- Poco costoso.
- Expandible.
- Flexible.
- Muy interoperable.

Fue desarrollado por Xerox en la década delos 70, operando a 2,94 Mbps, y se estandarizo abiertamente como *Ethernet Version 1* por el consorcio de *DEC, Intel* y *Xerox* (*DIX*). Más tarde fue redefinido como *Ethernet* II, y luego *el Institute of Electrical and Electronic Engineers* (*IEEE*) publicó un estándar de *Ethernet* funcionando a 10 Mbps (*802.3*).

IEEE 802.3 es una asociación profesional de trabajo en estándares sobre las capas físicas, control de acceso de medios (*MAC*) de la capa de enlace de datos de *Ethernet*. Define las redes locales de *Ethernet* y las áreas metropolitanas de red, mostrando compatibilidad con otros estándares más antiguos. *Ethernet* atiende a varias ampliaciones del estándar y variaciones de dichos estándar; sin embargo, todas estas normas usan especificaciones del control de acceso de medios y de la base de información de gestión (*MIB*). Los estándares *Ethernet* no necesitan

detallar todos los aspectos y funciones requeridos en un Sistema Operativo de Red (*NOS*), especifica únicamente las dos primeras capas del modelo *Open Systems Interconnection* (*OSI*):

- La capa física, constituida por el cableado y las interfaces.
- La capa de enlace, que se encarga de suministrar el direccionamiento local, detectar errores y monitorizar el acceso a la capa física.

El estándar original de *Ethernet 802.3* ha evolucionado con el paso del tiempo, soportando mayores velocidades de transmisión, distancias más largas, y nuevas tecnologías. La mayoría de categorías de *Ethernet* se organizan por su velocidad:

- *Ethernet* (10Mbps), regido por los estándares *802.3* (en cable coaxial) fino, *802.3ª* (cable coaxial grueso), *802.3i* (cable de par trenzado) y *802.3j* (fibra), dependiendo de la capa física utilizada (el tipo de cable).
- Fast *Ethernet* (100 Mbps), dirigido por el estándar *802.3u*. Introduce la habilidad de *autonegociar* (configurable en los aparatos), la velocidad y el tipo de operación *duplex* de la interfaz, que tratará de usar la máxima velocidad y *Full-duplex* si los dispositivos lo soportan.
- *Gigabit Ethernet*, conducido por los estándares *802.3ab* para cables de par tranzado y *802.3z* para la fibra.
- *10 Gigabit Ethernet*, guiado por los estándares *802.3an* para cables de par tranzado y *802.3ae* para la fibra.

Ethernet es una comunicación *baseband*, dedica toda su capacidad a trasladar la información de manera digital en un único canal de señal sin *multiplexar*. Para que se produzca la comunicación entre dispositivos debe utilizarse un cable cruzado, es decir la transmisión de un aparato debe conectarse con la recepción del otro, y viceversa.

La arquitectura *Ethernet* consiste en una red de conmutación de paquetes de acceso compartidos por varios terminales y difusión amplia, que emplea un medio (pasivo y sin control centralizado) administrado desde los equipos.

El protocolo *CSMA/CD* (*Carrier Sense Multiple Access with Collision Detect*) monitorea la portadora (tanto su detección como la de las posibles colisiones) y determina la operación *Half-duplex* o la operación *Full-duplex*. Cualquier dispositivo está autorizado a emitir en un momento dado, sin ninguna prioridad, ya que antes de comenzar la transmisión comprueba que no hay comunicación alguna, y si efectivamente así es, captura el canal enviando paquetes a intervalos no estándar, que evitan las transmisiones desde otros equipos. Sin embargo, ante una colisión de datos provenientes de dos equipos distintos, ambos interrumpen su emisión y esperan un periodo de tiempo (que varía según la frecuencia de las colisiones) antes de reiniciar su transmisión. Establece una serie de normas y limitaciones para la comunicación correcta [10]:

- Los paquetes de datos tienen un tamaño máximo.
- Hay un lapso de espera entre dos trasmisiones.
- Cada colisión de datos genera un tiempo de espera que dobla el de la anterior colisión, siendo el primero el intervalo necesario para enviar 96 bits.

El protocolo de los paquetes (*PDUs*) *802.3*, cambia ligeramente (en número de bits y en algunas partes del paquete añadidas) de cada en cada estándar de *Ethernet*, aunque su estructura mantiene en orden [10]:

- Un preámbulo, compuesto por 8 bytes que tienen el propósito de permitir que los equipos receptores sincronicen sus relojes con el mensaje para su lectura sin errores. El último bit es el delimitador de comienzo de *frame* (*SFD*) o paquete.
- Direcciones de origen y de destino, direcciones físicas de dispositivos (adaptadores de red) o *MACs*, que hacen referencia al adaptador al que va dirigido (*Recipient Address*) el mensaje y el equipo que lo emite (*Source Address*). Estas direcciones se representan mediante 6 bytes, 3 bytes de ellos suministrado por la *IEEE*, y son mundialmente únicas.
- Tipo de trama, que identifica el protocolo de alto nivel que se utiliza en la red *Ethernet*, en 16 bits.
- Bits de datos (desde 46 bytes a 1500 bytes).
- Chequeo de integridad (*FCS*), un *checksum* del *frame* de 32 bits. El equipo emisor del mensaje realiza un control cíclico redundante (*CRC*) de los datos e incorpora el valor en este campo, que será verificado por el dispositivo receptor mediante otro *CRC*, y en caso de fallar esta validación se solicita el reenvío del paquete.

Los interfaces independientes de medios (*MIIs*) proporcionan implementaciones opcionales y arquitecturas para las capas físicas (*PHY*), que a su vez codifican la estructura para la transmisión y decodifica la recepción de información con la modulación especificada para cada velocidad de funcionamiento, medio de comunicación y la longitud del enlace; también se encarga de la gestión y el control de protocolos, y el suministro de potencia sobre determinadas capas físicas de par trenzado.

Las capas físicas o conexiones físicas son establecidas entre nodos o dispositivos de infraestructuras como *hubs, switches* y *routers*, y se organizan dependiendo de la topología de red empleada:

- Topología de *bus*, en la que todos los *hosts* (anfitriones) comparten un único segmento físico para la comunicación. La información enviada por un *host* es recibida por todos los *hosts* del *bus*; sin embargo, solo será procesada si coincide con la dirección suministrada en la cabecera de datos de enlace.
  Para evitar la reflexión de la señal los extremos del *bus* deben estar terminados, por lo que añadir o eliminar anfitriones se vuelve más complicado.
- Topología de estrella, en la que cada anfitrión posee una conexión punto a punto a un *hub* o *switch*. A diferencia de la topología de *bus*, agregar y excluir *hosts* es muy fácil, y un fallo en el cableado afectaría solo a un anfitrión, no a toda la red.
  El *hub* o *switch*, usa mecanismos de conmutación y filtrado, analiza las direcciones de origen y destino de las transmisiones e inspecciona los puertos a los que se conectan los equipos. Además incrementa la velocidad de la conexión y evita colisiones, ya que al conocer el equipo receptor, solo se transmite el mensaje al dispositivo adecuado, y los otros puertos quedan disponibles para la comunicación.

A.2.2.2 Características

A grandes rasgos y atendiendo a los diversos estándares, *Ethernet* presenta las siguientes características, visibles en la tabla 21:

Tipo de Ethernet	Ancho de banda	Tipo de cable	Duplex	Distancia máxima
10Base-5	10 Mbps	Coaxial thicknet	Half	500 m
10Base-2	10 Mbps	Coaxial thinnet	Half	185 m
100Base-TX	10 Mbps	UTP Cat3/Cat5	Half	100 m
100Base-TX	100 Mbps	UTP Cat5	Half	100 m
100Base-TX	200 Mbps	UTP Cat5	Full	100 m
100Base-TX	100 Mbps	Fibra multimodo	Half	400 m
1000Base-T	200 Mbps	Fibra multimodo	Full	2 km
1000Base-TX	1 Gbps	UTP Cat5e	Full	100 m
1000Base-SX	1 Gbps	UTP Cat6	Full	100 m
1000Base-LX	1 Gbps	Fibra multimodo	Full	550 m
10GBase-CX4	1 Gbps	Fibra monomodo	Full	2 km
10GBase-T	10 Gbps	Twinaxial	Full	100 m
10GBase-LX4	10 Gbps	UTP Cat6a/Cat7	Full	100 m
10GBase-LX4	10 Gbps	Fibra multimodo	Full	300 m
10 Mbps	10 Gbps	Fibra monomodo	Full	10 km

Tabla 21. Tabla de características de los distintos estándares *Ethernet* [6]

A.2.2.3 Protocolos TCP/IP

TCP/IP es un conjunto de protocolos (normas) aplicados a procedimientos y formatos de mensaje, que permiten la comunicación remota entre equipos, determina meticulosamente cómo se transmite la información desde una máquina a otra.

La inmensidad de protocolos en una máquina puede parecer acarrear un problema; sin embargo, el tipo de protocolo indicado en el paquete, concretamente en el campo tipo de trama, permite que los paquetes se *autoidentifiquen*, lo que posibilita la usabilidad de numerosos protocolos en un mismo equipo entremezclados, sin errores o interferencias. Los

protocolos *TCP/IP* usan tramas *Ethernet autoidentificables* tanto en *IPv4* con un valor hexadecimal del tipo de trama de 0800 (77), como en *IPv6* con un valor hexadecimal del tipo de trama de 86DD (80). Aunque IPv6 incorpora diversas ventajas sobre *IPv4*, como la simplificación de tareas del *router*, mayor compatibilidad con redes móviles, mayor carga útil y un infinito direccionamiento, apenas se utiliza en comparación a *IPv4*. [10]

Los protocolos *TCP/IP* usados por *Ethernet* más comunes son [6]:

- Protocolo de Internet (*IP*), que suministra servicios de la entrega de tramas entre nodos.
- Protocolo de resolución de direcciones (*ARP*), que asigna direcciones de Internet (*IPs*) a direcciones físicas.
- Protocolo de control de mensajes de Internet (*ICMP*), que controla la transmisión de mensajes de error y control entre *routers* y *hosts*.
- Protocolo de control de transmisión (*TCP*), que facilita servicios de envío de flujos íntegros entre clientes.
- Protocolo de Protocolo de resolución de direcciones por réplica (*RARP*), que determina las direcciones físicas a direcciones de Internet.
- Protocolo de datagrama de usuario (*UDP*), que ofrece servicios de entrega de datagramas dudosos o poco importantes entre clientes. Es una alternativa poco fiable a *TCP*.
- Protocolo de transferencia de archivos (*FTP*), que proporciona servicios de nivel de aplicación para la transmisión de archivos.
- Telnet, que simula una terminal.
- Protocolo de abrir el camino más corto primero (*OPSF*), que posibilita la comunicación sobre rutas de estado del enlace entre *routers*.
- Protocolo de información de encaminamiento (*RIP*), que permite compartir información entre *routers* sobre rutas de vectores.
- Protocolo de puerta de enlace externa (*EGP*), que permite la comunicación de rutas entre *routers* externos.

La mayoría de estos protocolos operan en la capa de red del modelo *OSI*, excepto los protocolos *TCP* y *UDP*, que trabajan en las capas de sesión y transporte, y *Telnet* y *FTP*, que actúan en la capas de presentación y aplicación:

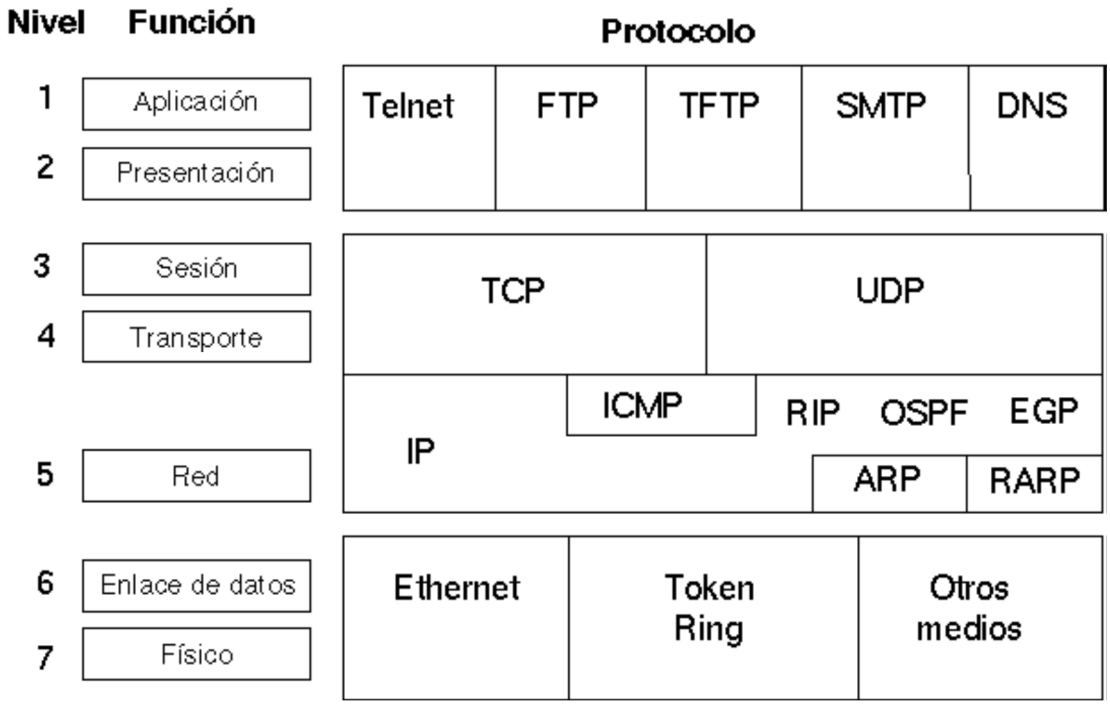


Figura 50. Protocolos comunes *Ethernet* en las capas *OSI* [6]

La comunicación entre capas *OSI* se fundamenta en que cada capa del sistema le añade información adicional de control a los datos, y cada capa del sistema receptor procesa la información de control. En la figura 51 se observan las capas del modelo *OSI*, su relación entre ellas y con la de otro sistema, aunque los protocolos *TCP/IP* sólo emplean las capas de aplicación, transporte, Internet (equivalente a la capa de red) e interfaz de red (constituida por la capa de enlace de datos y la capa física).

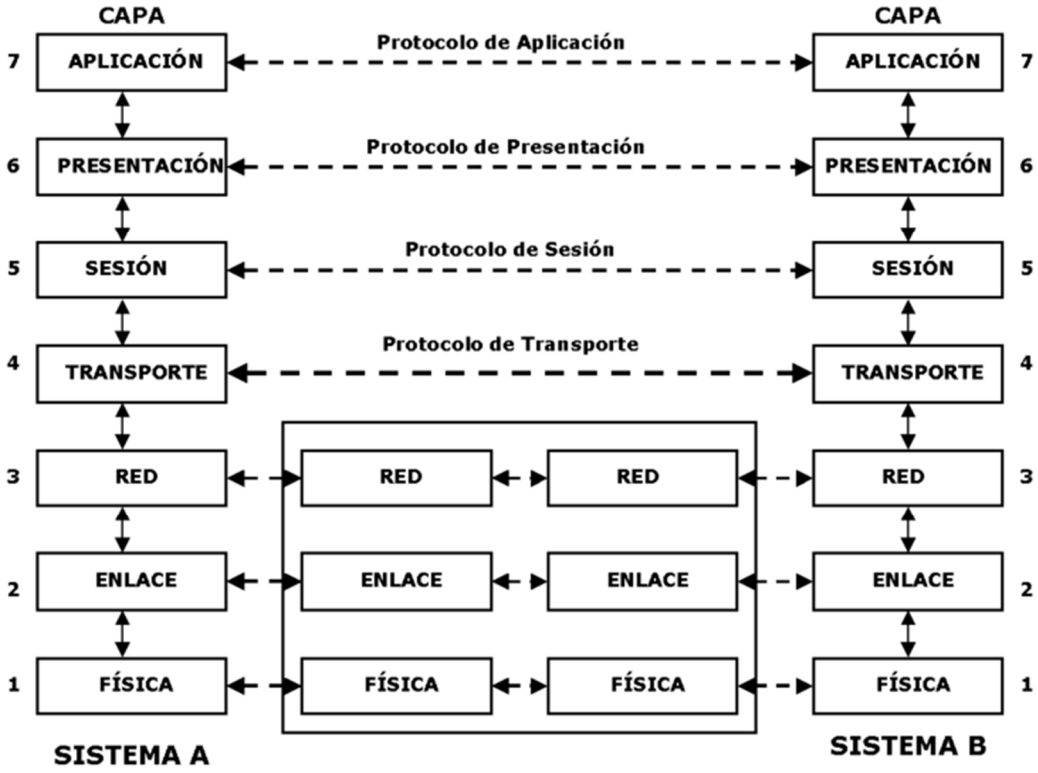


Figura 51. Relación *Ethernet* entre dos sistemas [6]

Las aplicaciones que utilizan *TCP/IP* en la transferencia de archivos emplean un grupo de protocolos de varias capas para realizar la comunicación, denominado *stack de protocolo*, de manera que el dato transmitido hacia abajo de capa en capa, hasta llegar a la capa física, donde se envía el paquete a través de la red.

Las aplicaciones de un sistema A envían mensajes o flujos de datos, mediante protocolos como *FTP, SMTP, Telnet*, etc., a uno de los protocolos de la capa de transporte de Internet, *TCP* o UDP, que reciben la información y la dividen en porciones más pequeñas (segmentos), agregan una cabecera con un número de secuencia a cada segmento, los envían a la capa de red o capa de Internet (*IP*) y se efectúa la suma de comprobación. [10]

La capa de red o capa de Internet elabora un paquete con los datos del segmento *TCP*, agrega un encabezado con las direcciones *IP* de origen y destino (los equipos que actúan como intermediarios hasta la máquina destinataria), a través del protocolo *ARP*, transmite el paquete a la capa de enlace de datos y se recalcula la suma de comprobación. La capa de enlace de datos transmite el paquete IP en el campo de datos de una trama de enlace de datos al *host* de destino. Así, el paquete está finalmente constituido por las cabeceras correspondientes a cada una de las capas implicadas y los datos procedentes de la aplicación:

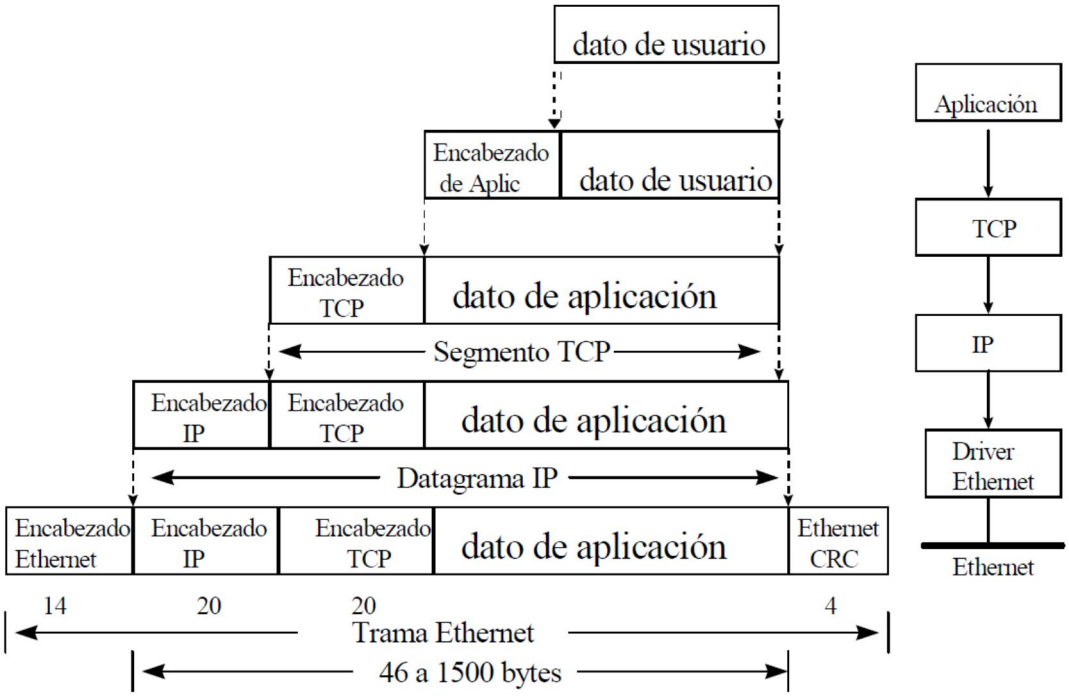


Figura 52. Trama *Ethernet* [6]

Si el equipo que recibe el paquete resulta ser un intermediario, el proceso desde que la capa de IP elabora el paquete se repite hasta que logre llegar hasta el sistema de destino B.

El sistema destinatario B, que recibe las tramas en la capa física, realiza el proceso inverso, cada capa elimina la información del encabezado correspondiente hasta que los datos regresan a la capa de aplicación. La capa de enlace de datos retira el encabezado *Ethernet* y suministra el datagrama a la capa de red o capa de Internet (*IP*), donde el protocolo de internet verifica que la suma de comprobación coincide con la calculada, de ser así quita el encabezado *IP* y entrega el paquete a la capa de transporte, sino se desecha el paquete. En la capa de transporte, el *TCP* verifica el número de secuencia, ordena los paquetes y valida la suma de comprobación, y en caso de estar intactos, excluye el encabezado *TCP* y transmite los datos a la capa de aplicación, en caso contrario elimina el segmento. Finalmente, la capa de aplicación del sistema B, recibe el flujo de datos.

A.2.3 Conversores e interfaces de comunicación

RS-232 y *RS-485* son estándares de comunicación diseñadas e implementadas para unificar el lenguaje de comunicación entre distintas máquinas y dispositivos que conforman un sistema, y siguen presentes en la mayoría de los sistemas; sin embargo, surge la necesidad de reemplazados por el estándar *Ethernet* unido a los protocolos *TCP/IP* para conectarlos empleando la misma estructura de red.

Existe una tendencia mundial a utilizar la red *IP* como exclusiva para todas las comunicaciones, lo que supone una reducción considerable en los costes de infraestructura e integración. Ante

la ausencia de *hardware* dedicado a la comunicación mediante protocolos *TCP/IP*, los conversores aumentan las capacidades de uso a cambio de una inversión muy reducida.

La conversión a *TCP/IP* consiste en transmitir los datos provenientes de las normas serie a la interfaz de Internet, de manera que se ajustan los niveles eléctricos y se encapsula la información en un paquete, para posteriormente transmitirlo por la red. Para llevar a cabo este proceso, los conversores *TCP/IP* están constituidos por *microcontroladores* potentes, debido a la cantidad de información y el protocolo que hay que seguir para la comunicación. El *microcontrolador* recoge los datos del medio serie, reconvierte el protocolo de datas, elabora el protocolo *TCP/IP* y manipula el chip de red (encargado de adaptar las características eléctricas), incorporado cada vez más en el propio *microcontrolador*.

A.2.3.1 Dispositivo XPort de Lantronix

XPort es una solución compacta e integrada que permite la comunicación de cualquier dispositivo serie mediante *Ethernet*. Elimina la complejidad del diseño de la conectividad a la red, incorporando todo el *hardware* y *software* necesarios en un único elemento embebido [14]:

- Conexión *Ethernet 10Base-T* y *100Base-TX*.
- Sistema operativo ro*bus*to en tiempo real (*RTOS*).
- *Web Server*.
- *Stack de protocolo TCP/IP*
- Alertas e-mail.
- Encriptación *AES* de 256 bits.

Posee un procesador de red *DSTni™ SoC*, que integra *10/100 MAC/PHY* y 256 KB de *SRAM*. Sus características están construidas en un servidor web para remotamente comunicar, configurar, controlar y/o resolver problemas.

A.3 Tecnologías embebidas programables

La implementación de soluciones tecnológicas electrónicas entraña multitud de posibles desarrollos. Los circuitos integrados presentan una alternativa idónea para el diseño de cualquier dispositivo electrónico por su tamaño, prestaciones y coste. Un circuito integrado (*CI*) es un conjunto de circuitos electrónicos concentrados en un área muy pequeña de material semiconductor (generalmente silicio). En la actualidad se pueden diferenciar varias clasificaciones atendiendo a la manera en que se realiza el diseño de los sistemas electrónicos digitales [16]:

- *Standard off-the-shelf integrated circuits*. Son circuitos integrados monolíticos estandarizados, es decir con unas características bien determinadas. Se Clasifican en:
    - Circuitos integrados monolíticos de función fija. Diseñados para cumplir con una o varias funciones específicas, se subdividen según el tipo de aplicación:
        - De aplicación general, son aquellos se usan en la elaboración de múltiples circuitos y sistemas, y desempeñan funciones tanto únicas como diversas. Los multifuncionales cuentan con un conjunto de variables binarias de entrada, que posibilitan escoger las funciones Por ejemplo: Como multifuncional *ALU74LS181* y *74LS00* de función única.
        - De aplicación específica. Estos dispositivos ejercen bien una función complicada definida o en su lugar un ámbito complejo de un sistema digital superior. Como modelos, *UARTS* o interfaces para el caso de realizar una parte difícil de un sistema más grande, o un mando de televisión como tarea complicada.
    - Circuitos integrados monolíticos de función programable. Desempeñan una función que puede ser modificada. Atendiendo a su arquitectura, es decir, los componentes y las relaciones existentes entre ellos, se organizan en dos grupos:
        - De arquitectura fija. Poseen un *hardware* compuesto por elementos unidos entre sí, inalterables por el usuario. Según el tipo de sistemas que ofrecen, se subdividen en:
            - *Combinacionales*, aquellos formados por múltiples compuertas interconectadas y definidos a través de una tabla de valores de verdad. No tienen en cuenta el estado anterior del sistema, la salida es únicamente dependiente de la entrada. Por ejemplo: *RAM, PLA, PAL*, etc.
            - *Secuenciales*, en los que respuesta del sistema depende del estado anterior de este. Emplean memorias de acceso aleatorio para almacenar el estado o los estados anteriores a un momento dado. Suelen estar constituidos por *CPU*s, todos los elementos de un ordenador, *DSPs*. Por ejemplo: microprocesadores, *microcontroladores* y *PICs*, respectivamente para los componentes antes citados.

- De arquitectura configurable. El *hardware* y la relación entre los componentes del mismo, se transforman para ajustarse a cambios en las especificaciones, a través de la alteración de unas variables binarias. Se dividen en varios tipos de circuitos digitales de acuerdo al tipo de organización:
  - ❖ Con recursos de organización matricial, integrado por dispositivos lógicos programables (*PLD*). Se catalogan en:
    - o BLPD, compuestos por una *PAL* realimentada mediante un circuito lógico que establece una *Macrocell*. Las *PLDs* básicas presentan limitaciones que se solucionan empleando *PLDs* avanzadas.
    - o *APLD*. Se trata de la misma estructura que un *BLP*, pero, los *DLP* poseen recursos dedicados a que las *Macrocell* compartan recursos de la *PAL*, y están abastecidos de varias matrices de interconexión
    - o *CPLD*, son *APLD* mejorados mediante la distribución de recursos de interconexión, lo que les convierte en unos circuitos digitales muy flexibles y programables.
  - ❖ Con recursos de interconexión distribuidos, constituidos por *Field Programmable Gate Arrays* (*FPGA*). Son conjuntos configurables de puertas constituidos por bloques lógicos de entrada y salida, recursos de interconexión y bloques lógicos internos. Atendiendo a la disposición de estos últimos se aprecian tres organizaciones:
    - o *Terraces*, en la que los bloques lógicos internos se ordenan en filas, entre las que se posicionan recursos de interconexión horizontal en canales. Presentan estructuras de granularidad fina o gruesa, dependiendo de la complejidad de los bloques lógicos internos, interconectados mediante recursos de interconexión horizontal y vertical. Los recursos de interconexión internos horizontales son más numerosos y se usan para unir entradas y salidas de los bloques; sin embargo, los recursos de interconexión verticales proporcionan comunicación entre canales horizontales y propagan señales generales.
    - o *Manhattan*, en la que los bloques lógicos internos se distribuyen en filas y columnas demarcadas por los recursos de interconexión horizontales y verticales.
    - o *Sea of Gates*, consiste en un *array* de bloques simétrico de dos dimensiones. Cada bloque es un conjunto de celdas idénticas en una distribución de m filas y m columnas, por lo que lo convierte en un diseño poco costoso, veloz en la entrega y en la

creación de prototipos; sin embargo, requiere de más espacio, por ende no se utiliza en producciones de volúmenes cuantiosos.

Las *FPGAs* también se pueden encasillar en tres clases considerando el tipo de elemento de configuración:

- *Antifusibles*, representan la antítesis del fusible, son dispositivos programables para pasar de un circuito abierto a una baja impedancia. Generalmente estas *FPGAs* están compuestas por raíces de elementos lógicos configurables con canales de interconexión entre ellos. Solo facilitan una programación.
- Transistores *MOS* de puerta flotante, en los que la puerta está aislada eléctricamente. Ofrecen configuraciones múltiples.
- Celdas de memoria estática, que se ocupan de guardar el programa que representa el diseño del usuario, implementan lógica (como *LUTs*) en la que las entradas de la función dominan las líneas de direccionamiento. Permite la configuración de las *FPGAS* numerosas veces.

- *Application Specific Integrated Circuits* (*ASICs*) [34]. Son circuitos específicos que implementan un sisetma electrónico para una aplicación determinada por el usuario, por lo que para su diseño son descritos a diferentes niveles siguiendo una metodología *Top-Down*. En primer lugar es necesario definir el sistema como una caja negra con sus respectivas variables de entrada y salida, para a continuación detallar las funciones que realizará y posteriormente ampliar todas estas funciones en bloques formados por las unidades mínimas (componentes) y su relación entre ellas. Según el nivel alcanzable en el diseño se clasifican en dos grandes grupos:
  - *Full custom*, son aquellos circuitos integrados monolíticos diseñados totalmente a medida, hasta el nivel de elegir las características y el conexionado de transistorores, lo que les confiere un bajo consumo de potencia y elevados rendimientos y velocidades. Sin embargo, supone un sistema costoso, por lo que solo se justifica su aplicación en series de grandes volúmenes o en la búsqueda de la optimización del circuito.
  - *Semi custom*. Son circuitos integrados en los que bloques funcionales predefinidos (*Cells* o celdas) son seleccionados para cumplir el propósito de la aplicación, por lo que solo el diseñador solo determina decisiones lógicas. Se pueden organizar considerando la complejidad de las celdas y la relación entre ellas en:
    - *Mask Programmable Gate Arrays* (*MPGA*), formados por componentes distribuidos regularmente y amplificadores conectados a terminales externos, desempeñan una función específica. Por ejemplo un conjunto de puertas universales (*NAND*, *OR*, etc.) interconectadas para cumplir una función, o un grupo de transistores que constituyen un bloque funcional como multiplexores, contadores, etc.

- *Standard Cell*, son celdas estandarizadas, celdas agrupadas en bloques funcionales, cuya actividad está garantizada por elementos como registros, decodificadores, multiplexores, etc. Las celdas y sus relaciones no están predeterminadas, lo que aumenta el aprovechamiento de las tarjetas de silicio de circuito impreso.

Aunque representan casi la totalidad de los tipos circuitos integrados monolíticos usados en el diseño de sistemas digitales en la actualidad, en diversas aplicaciones es necesario diseñar sistemas destinados a lograr tareas complejas que requieren de la utilización de varios circuitos anteriormente citados, aquellos que implementan un sistema digital con circuitos que realizan funciones antes mencionadas. Esta clase de circuitos se denominan circuitos integrados mixtos.

A.3.1 *Microcontrolador*

Un *microcontrolador* es un circuito integrado de pequeño tamaño que en su interior está constituido por una unidad central de procesamiento (*CPU*), puertos de entrada y salida, unidades de memoria (*RAM* y *ROM*) y periféricos, interconectados entre ellos. Representa la totalidad de los elementos básicos de un ordenador personal embebido y es capaz de gobernar objetos, procesos o eventos, programados por el usuario e introducidos mediante un programador.

Frecuentemente se suelen entremezclar los términos *microcontrolador* y microprocesador y pueden dar lugar a confusión, pero la diferencia entre ambos se encuentra en que un *microcontrolador* se trata de un sistema cerrado (con todos los componentes de un ordenador) y un microprocesador es un sistema abierto, necesita una conexión a periféricos (dispositivos de entradas/salidas y memoria). Aunque a simple vista el *microcontrolador* parece una opción más ventajosa por su bajo coste y la incorporación de periféricos, el microprocesador no está tan limitado por eximir la responsabilidad de albergar los elementos básicos de un ordenador en su interior, por lo que el uso de uno u otro depende de la aplicación a la que estén dedicados.

Actualmente se pueden encontrar multitud de *microcontroladores* con diversas características, cualquier sistema que realice medidas, almacene, controle, calcule o presente información es un firme candidato a incorporar un *microcontrolador*. La clasificación más adecuada teniendo en cuenta las características de los *microcontroladores* es la distribución de acuerdo a la longitud de palabra:

- Con dimensiones de 4 bits, la gama más baja y más económica.
- Con tamaño de 8 bits, que actualmente dominan el mercado.
- Con capacidad de 16 bits, con prestaciones y costes superiores a los dos anteriores.
- Con una longitud de 32 bits, que cuentan con el mejor rendimiento y poco a poco inclinan el mercado hacia una migración a este tipo de controladores por la demanda de dispositivos de alto desempeño.

A.3.1.1 Arquitectura

Existen dos arquitecturas predominantes en el ámbito de los *microcontroladores* (*Von Neumann* y *Harvard*), ambas basadas en cuatro subsistemas:

- Memoria
- Entradas/Salidas (*I/O*)
- Unidad aritmético lógica (*ALU*)
- Unidad de control (*CU*)

La unidad de control y la unidad aritmético lógica operan juntas para crear la unidad central de procesamiento, donde residen los registros (memorias de alta velocidad y escasa capacidad que contienen instrucciones y datos).

Las instrucciones de ejecución con cualquier arquitectura se fundamentan en el mismo ciclo Obtener ➔ Decodificar ➔ Ejecutar. Las instrucciones son llevadas desde la memoria de acceso aleatorio (*RAM*) a los registros de instrucciones, donde la unidad de control decodifica la instrucción y la envía a la *ALU*, que a su vez realiza la operación apropiada en los datos y transmite el resultado de vuelta a la unidad de control para su almacenamiento. Sin embargo, la diferencia entre estas arquitecturas radica en la integración o separación de las instrucciones de programa y los datos en la memoria. [8]

A partir de estas arquitecturas surgieron arquitecturas más modernas (como *RISC* y *CISC*) e híbridos entre ellas que siguen satisfaciendo las demandas de mayor velocidad en el procesamiento de datos.

A.3.1.1.1 Arquitectura *Von Neumann*

Es la arquitectura más usada en los sistemas con microprocesadores. La unidad central de procesamiento está conectada a una unidad de memoria (generalmente constituida solo por *RAM*) que alberga las instrucciones de programa y los datos y se comunica por medio de un único sistema de *bus*es. Las dimensiones de la unidad de datos o instrucciones está determinado por el tamaño del *bus* que relaciona la memoria con la *CPU*, es decir, un microprocesador de 8 bits con un *bus* de 8 bits manipula datos e instrucciones de programa de una longitud proporcional a un byte, lo que le obliga a acceder más veces a la memoria si las instrucciones o datos ocupan más de un byte. [8]

La singularidad del *bus* confiere a la arquitectura Von Neumann una velocidad de operación más lenta, pues no permite solapar tiempos de acceso a datos e instrucciones al no posibilitar el acceso simultáneo a unos y a otros.

A.3.1.1.2 Arquitectura Harvard

Esta arquitectura es usada en los *microcontroladores PIC*, una familia de *microcontroladores* de tipo *RISC* (*Reduced Instruction Set Computer*). Posee una unidad central de procesamiento conectada a dos memorias (una de datos y otra de instrucciones) a través de dos *bus*es independientes, lo que permite la superposición de los tiempos de acceso a los datos e instrucciones y la diversidad de longitudes y contenidos en la misma dirección, que incrementa la velocidad significantemente. Cada operación de acceso a instrucciones es capaz solaparse con el acceso a datos y además las instrucciones pueden ocupar una sola posición de memoria de programa, que resulta en un programa breve y rápido. [8]

A nivel funcional presenta un mejor rendimiento; sin embargo, es necesario más *hardware* para dos *bus*es y dos chips de memoria o una memoria de doble puerto.

A.3.1.1.3 Arquitectura *RISC* y *CISC*

Las arquitecturas *RISC* y *CISC* se fundamentan en registros, pero son dos corrientes que han seguido caminos diferentes. [46]

Un procesador *CISC* (*Complex Instruction Set Computer*) se apoya en la microprogramación, cada instrucción es descifrada por un microprograma situado en una memoria, internamente en el circuito integrado del procesador, y ejecutada con microinstrucciones que residen en una *ROM* del interior del chip. Como resultado permite ejecutar cientos de instrucciones complejas distintas, ofreciendo una gran versatilidad y recursos.

Por el contrario, los procesadores *RISC* (*Reduced Instruction Set Computer*), basados en la arquitectura *Von Neumann*, solo ejecutan instrucciones sencillas establecidas por *hardware* en la *CPU*, lo que provoca la desaparición del *microcódigo* y la decodificación de instrucciones complejas. Permiten la segmentación (los cálculos son registrados con el reloj cada cierto tiempo para que la ruta crítica disminuya), el paralelismo en la ejecución de instrucciones (varias instrucciones a la vez) y reducen los accesos a la memoria (las instrucciones de carga y almacenamiento acceden a la memoria de datos).

Esto les convierte en dispositivos más económicos, más simples y capaces de trabajar a una frecuencia más elevada. Aunque procesan menos instrucciones lo compensan a través de una ejecución mucho más veloz.

La existencia de estas arquitecturas tan dispares puede desembocar en la decisión de utilizar una u otra según la aplicación de cada caso, pero con el tiempo se han desarrollado procesadores híbridos, que consisten en procesadores *RISC* con recursos propios de los procesadores *CISC*, o viceversa. Así, el procesador se encarga de las instrucciones simples, que son acomodadas al proyecto del chip por medio de un decodificador que transforma las instrucciones complejas en instrucciones simples.

A.3.1.1.4 Arquitectura interna

Un *microcontrolador* mantiene unas características básicas y una estructura fundamental muy universales. Está compuesto por varios bloques funcionales interconectados para cumplir una tarea específica. Esencialmente estos tres bloques son:

- Unidad central de procesamiento (*CPU*)
- Memoria
- Unidades de entrada y salida

A.3.1.1.4.1 Unidad central de procesamiento (CPU)

Es la parte principal de cualquier *microcontrolador*, a menudo usada desde un procesador existente. La arquitectura básica de esta unidad consiste en una ruta de datos que ejecuta las instrucciones, y la unidad de control que básicamente le indica a la ruta qué realizar.

En el núcleo de la *CPU* se encuentra la unidad aritmético lógica (*ALU*), que se encarga de realizar los cálculos (*AND, ADD, INC*, etc.). Varias líneas de control seleccionan que operación en los dos datos de entrada (obtenidos desde los registros o la memoria) debe ejecutar la *ALU*, que devuelve el resultado de la operación como una salida, que se almacena en los registros o en la memoria, y la naturaleza del resultado (*Zero, Negative, Overflow* o *Carry*) que se guarda en el registro de estado. [8]

Los registros funcionales de la *CPU* residen en el fichero de registro, que consiste bien en un conjunto de registros de propósito general usados para almacenar entradas o resultados de operaciones, o en su lugar está constituido por registros dedicados (acumulador, registro de índice, etc.). Los datos grabados en los registros disponen de un acceso más rápido que las memorias, por lo que se suelen utilizar cuando es posible.

La *CPU* usa una porción de memoria consecutiva en su espacio de datos, la pila, para almacenar las direcciones de retorno y posiblemente los contenidos del registro, durante subrutinas y llamadas de rutina de servicio de interrupción. La *CPU* incluye el nivel actual de la pila en un registro especial (*Stack Pointer*) que apunta a la parte superior de la pila y suele empezar al final de la memoria de datos, de manera que si almacenamos un dato (*PUSH*) en la pila, el valor del *Stack Pointer* decrece, y si extraemos algo el valor del *Stack Pointer* aumenta.

La *CPU* constantemente ejecuta instrucciones de programa, excepto en situaciones excepcionales (*HALT, Reset*…), y necesita de la unidad de control para determinar qué operación será la siguiente y qué ruta de datos en consecuencia. Para ello, un registro especial, el contador de programa (*PC*) lleva a cabo el almacenamiento de la dirección de la siguiente instrucción de programa. La unidad de control carga esta instrucción en el registro de instrucciones (*IR*), decodifica la instrucción y configura la ruta de datos (proporciona las

entradas apropiadas para la *ALU*, la operación adecuada y se asegura de que el resultado es grabado en el destino correcto) para ejecutarla.

A.3.1.1.4.2 Memoria

Funcionalmente, desde la perspectiva de un programador, existen tres tipos importantes de memoria clasificables en:

- Fichero de registro, que acostumbra a ser una memoria pequeña embebida en la *CPU* y es utilizada para almacenar valores temporales con los que la *CPU* opera.
- Memoria de datos, una memoria externa a la *CPU* y más extensa que el fichero de registro, que se emplea para guardar datos durante más tiempo.
- Memoria de instrucciones, una memoria externa y relativamente grande, integrada en la unidad de control de los *microcontroladores*. En la arquitectura de *Von Neumann* es físicamente la misma memoria que la memoria de datos.

Sin embargo, los diseñadores de *hardware* o chips prefieren distinguir los distintos tipos de memoria atendiendo a las características físicas de los componentes, en memoria volátil y memoria no volátil.

A.3.1.1.4.2.1 Memoria no volátil

Las memorias no volátiles conservan la información guardada aun cuando no existe alimentación, en contraste su velocidad es mucho menor y son más complicadas. Hay numerosos tipos de memoria no volátil con diversas características que les permiten una mejor adaptación a unas aplicaciones u otras. En el mercado actual se usan mayoritariamente *ROM, PROM, EPROM, EEPROM, FLASH* y *NVRAM*.

A.3.1.1.4.2.1.1 Memoria *ROM*

La *Read Only Memory*, como su nombre indica, sólo puede ser leída, por lo que la información que contiene debe ser grabada previamente por el fabricante. Esto lo convierte en una opción justificable únicamente para grandes volúmenes de producción.

Generalmente, la celda de memoria consiste en un único transistor en el que el umbral de tensión define el estado lógico 1 o el estado lógico 0. Durante la lectura se programa tensión en la puerta del transistor de la celda y dependiendo del umbral de tensión establecido, el transistor conducirá o no la corriente. Después, un amplificador transforma esta corriente en un 1 o un 0.

Un circuito integrado *Mask-ROM* (*MROM*) es una clase de memoria *ROM* muy empleada, está encapsulada en un chip compuesto por diversas capas (creadas por máscaras) que determina la función del chip atendiendo a su geometría (contactos de metal para conectar el transistor a la línea, un implante de canal para crear un transistor de enriquecimiento o empobrecimiento,

óxido de puerta, etc.). La fabricación dispone las celdas de memoria en forma de matriz y en únicamente una capa se crean las conexiones entre filas y columnas, reflejando los datos a ser almacenados en la *MROM*.

A.3.1.1.4.2.1.2 Memoria *PROM*

La memoria *PROM* (*Programmable Read Only Memory*) supone una alternativa programable por el usuario a la memoria *ROM*. Consiste en matrices de celdas de memoria, cada una con un fusible de silicio inicialmente intacto y leído como estado lógico 1. Al hacer pasar un pulso de corriente elevada el fusible se destruye y la lectura del estado lógico se convierte en 0.

Los *microcontroladores OTP* (*One Time Programmable*) están diseñados con memoria de instrucciones *PROM*, por lo que no son adecuados para el desarrollo, pero una vez finalizado se transforman en una opción apropiada para producciones de volúmenes intermedios.

A.3.1.1.4.2.1.3 *EPROM*

Erasable Programmable Read Only Memory ofrece la ventaja de seguir cambiando la información contenida en la memoria tras su grabación, ya que ésta no es destructiva. Los datos se almacenan en la puerta de los transistores *FET* (*Field Effect Transistor*), que está completamente aislada del resto del circuito, a través de un proceso llamado inyección de avalancha. Presentan una vida útil de unos diez años, hasta que la puerta deja de esta completamente aislada, aunque si los chips de silicio *EPROM* son expuestos a luz ultravioleta se borran sus contenidos en minutos, lo que les otorga la cualidad de grabación múltiple (incorporan una pequeña ventana normalmente tapada que expone al chip a la luz ultravioleta).

A.3.1.1.4.2.1.4 *EEPROM*

Las memorias *EEPROM* (*Electrically Erasable and Programmable ROM*) comparten las ventajas de las memorias *EPROM*; sin embargo, eliminan el borrado de memoria mediante luz ultravioleta para incorporarlo a través de un voltaje elevado. Este voltaje puede superar la alimentación externa del chip y es generado mediante bombas de carga.

No obstante, tiene unos ciclos de escritura/borrado limitados, en torno a 100000, es costosa y no conserva la información indefinidamente. Es entonces una elección idónea para parámetros de calibración.

A.3.1.1.4.2.1.5 *FLASH*

La memoria *Flash* es una variante de la memoria *EEPROM* en la que no se permite borrar exclusivamente una dirección, solo bloques enormes de memoria o la memoria entera y un número de ciclos determinado (desde 1000 a 10000). Pese a que esto evidencia un inconveniente, reduce significativamente el precio y la velocidad de borrado es mucho mayor, por lo que constituye una fuerte alternativa a la *EEPROM*.

A.3.1.1.4.2.1.6 *NVRAM*

Non-Volatile RAM combina las ventajas de las memorias volátiles y no volátiles, por medio de pequeñas baterías internas en dispositivos *SRAM* (la batería proporciona energía a la memoria cuando se retira la alimentación) o la mezcla de una *SRAM* con una *EEPROM* (los datos son leídos y escritos en la *SRAM* durante la operación y cuando la alimentación desaparece la información se copia a la *EEPROM*).

A.3.1.1.4.2.2 Memoria volátil

Se trata de una memoria cuyos contenidos son eliminados cuando la alimentación del sistema, o en este caso del *microcontrolador*, desaparece. Aunque supone una desventaja frente a una memoria capaz de permanecer en cualquier momento, la memoria volátil es más veloz y menos costosa. Se organiza principalmente, por popularidad de uso, en memoria *SRAM* y memoria *DRAM*, cada una con sus ventajas e inconvenientes, lo que fomenta su coexistencia en *microcontroladores*.

A.3.1.1.4.2.2.1 *SRAM*

La *Random Access Memory* consiste en un *array* de celdas, cada una capaz de guardar un bit de información, a través de *flip-flops* (compuestos por seis transistores), manejada a través de los siguientes elementos:

- *Data In*, una entrada que proporciona a la celda el valor a ser almacenado.
- *Data Out*, una salida que muestra el valor almacenado en la celda.
- *Read/Write*, una entrada cuyo valor indica el tipo de acceso (0 para escribir el valor de *Data In* y 1 para devolver el valor almacenado por *Data Out*).
- *Cell Select*, una entrada que habilita o incapacita la celda, de manera que cuando la celda está inhabilitada (estado lógico 0) no es posible almacenar un valor ni retornar el valor guardado (*Data Out* muestra un estado de alta impedancia).

Estas celdas agrupadas conforman un chip que mantiene una estructura de entradas y salidas similar a las celdas individuales:

- *Address Lines*, que permite seleccionar celdas de memoria en concreto.
- *Data In*, son aquellos valores que se desean almacenar.
- *Data Out*, que facilitan los valores de las celdas elegidas.
- *Chip Select*, que capacita o incapacita el chip.
- *Read/Write*, que posibilitan una clase de acceso u otra.

A.3.1.1.4.2.2.2 *DRAM*

La *Dynamic Random Access Memory* está constituida también por celdas de memoria agrupadas en una estructura semejante (matrices de celdas), pero en comparación a la memoria *SRAM*, reduce el número de transistores usados en los *flip-flops* a uno (usado para seleccionar el tipo de acceso) e incorpora un condensador (almacena en forma de carga el estado lógico), lo que resulta en un mejor aprovechamiento del espacio y por tanto posibilita una mayor capacidad de almacenar datos. En contraposición, la memoria *DRAM* es más lenta y necesita un refresco del estado de las celdas continuo.

A.3.1.1.4.2.3 Unidades de entrada/salida

La habilidad de monitorizar y controlar el *hardware* es la principal característica de los *microcontroladores*, llevada a cabo a través de transferencias de datos que constituyen operaciones de entrada y salida, entre los circuitos externos y el microprocesador. Cada puerto de E/S puede ser una entrada, una salida o más comúnmente bidireccional, es decir, permite tanto la lectura como la escritura de datos desde el interior del *microcontrolador*. Además, algunos puertos tienen funciones alternativas seleccionables por el programador de la aplicación.

Los puertos de entradas y salidas, exceptuando funciones especiales, son digitales, de manera que el *microcontrolador* opera con estados lógicos (nivel alto 1 o nivel bajo 0). El comportamiento de cada puerto es determinado por tres registros:

- *Data Direction Register* (*DDR*), que establece la configuración del puerto bidireccional a entrada o salida, mediante un bit programable que puede ser leído a posteriori para comprobar su estado. Ante un reinicio o una incompatibilidad, el bit de *DDR* inicializa el puerto como entrada.
- *Port Register* (*PORT*). Controla el nivel de tensión de los puertos de salida, de forma que si el bit del registro *PORT* está en el estado lógico, el nivel de tensión será alto, y si el bit del registro *PORT* se encuentra en estado lógico 0, el nivel de tensión será bajo. Su lectura, en puertos de salida retorna el valor programado, y en puertos de entrada devuelve bien el valor establecido, o en su lugar el estado lógico del puerto.
- *Port Input Register* (*PIN*), es un registro de solo lectura que contiene el estado actual de los puertos, tanto de entrada como de salida.

A.3.1.2 Recursos especiales

Los fabricantes de *microcontroladores* orientan sus modelos a unas aplicaciones determinadas, para lo que reducen o amplifican algunas prestaciones e incluyen una serie de recursos complementarios a la arquitectura básica que consolidan la potencia y flexibilidad del *microcontrolador*. Entre los complementos más comunes se encuentra los temporizadores, perros guardianes, protecciones ante fallo de alimentación, estado de reposo o de bajo consumo, conversores *A/D* y *D/A*, comparadores analógicos, *PWMs*, puertos de comunicación e interrupciones.

A.3.1.2.1 Temporizadores

El módulo temporizador se ha convertido en una parte importante del *microcontrolador*, hasta tal punto que la mayoría de *microcontroladores* incorporan uno. Los temporizadores se utilizan para una inmensa variedad de tareas relacionadas con el control del tiempo, pero el uso más común es como contador, llevar la cuenta de acontecimientos que ocurren en el exterior, disparar interrupciones tras un número de ciclos de reloj, capturar el valor de un instante de tiempo o generar modulaciones de ancho de pulso.

Por defecto, en su aplicación como reloj interno, el temporizador aumenta con cada pulso del sistema (un oscilador externo). En su uso como contador, se incrementa o disminuye con cada pulso hasta que se desborda (sobrepasa el límite del valor de conteo) y el conteo es reiniciado de nuevo.

A.3.1.2.2 Perro guardián

Wacthdog o perro guardián se emplea para monitorizar la ejecución del *software*, consiste en la habilitación de un temporizador que comienza una cuenta descendente y al llegar a 0 provoca un reinicio automático del sistema. Para evitar el *reset* ante una ejecución normal del *microcontrolador* el *software* debe impedir que el temporizador alcance el valor 0, reanudando su cuenta.

Es un mecanismo dedicado mayormente a la prevención de bloqueos o a la verificación de que ciertas posiciones en el código programa se han sobrepasado en un periodo de tiempo, por lo que cuenta con su propio oscilador interno.

A.3.1.2.3 Protección ante fallo de alimentación

La protección ante una alimentación inestable es importante, en los *microcontroladores* el *Brown-out reset* (*BOR*), reinicia siempre que la tensión de suministro sea inferior a un voltaje

mínimo. De esta eluden se evaden comportamientos imprevistos del *microcontrolador* que puedan dañar el sistema.

A.3.1.2.4 Estado de reposo o de bajo consumo

El consumo de energía es una característica de cualquier sistema y determinante en aplicaciones dotadas de una alimentación suministrada por baterías. Para reducir este consumo existen diversas técnicas:

- Reducción de la frecuencia del reloj, ya que el uso de la energía es proporcional a la frecuencia. El diseñador puede perfectamente disminuir la frecuencia de trabajo para lograr un menor consumo; sin embargo, también es posible empequeñecer la frecuencia en momentos en los que el *microcontrolador* no necesite cumplir con unas limitaciones de tiempo ajustadas.
- Restricción del voltaje, el consumo de energía es proporciona al cuadrado de la tensión, así que una atenuación del voltaje de trabajo provocará que se mitigue el consumo, pero una excesiva disminución de la tensión de trabajo (por debajo del límite marcado) provoca comportamientos inesperados en el *microcontrolador*.
- Apagado de los módulos sin usar, consiste en mantener activos solo los módulos del *microcontrolador* que se encuentren en funcionamiento, mientras que el resto pueden ser apagados hasta que sean necesitados. Este método es empleado en los modos de reposo de los *microcontroladores*, que pueden llegar a apagar todos los módulos internos incluido en oscilador externo, con lo que se congelan los circuitos asociados, dejando al *microcontrolador* en un estado de inactividad y ahorro de energía, del que se escapa al activarse una interrupción ocasionada por un acontecimiento inesperado.
- Diseño optimizado, mediante una selección adecuada de los componentes que forman el *microcontrolador* se puede lograr un gran ahorro de energía.

A.3.1.2.5 Conversor *A/D*

El conversor analógico digital es una herramienta potente para reproducir el valor analógico en forma digital. Se establece una relación entre la entrada analógica y el resultado digital atendiendo a una resolución, que presenta un valor máximo (valor binario) para la información de entrada. De manera que el valor analógico correspondiente al bit menos significativo representa la resolución (V/bit) del conversor.

A.3.1.2.6 Conversor *D/A*

El conversor digital analógico es una utilidad que transforma una señal digital (valor binario) en una señal analógica. Realiza el paso inverso al conversor analógico digital. Dada una resolución (V/bit) preestablecida de acuerdo al valor máximo representable por el *microcontrolador* y un valor digital binario (de tantos bits como marquen las características de cada *microcontrolador*) es posible reproducir la señal analógica correspondiente a este valor binario.

A.3.1.2.7 Comparador analógico

Un comparador analógico posee dos entradas analógicas, ambas externas o al menos una de ellas y la otra generada internamente, y una salida digital que muestra el resultado de comparar el valor de la primera entrada con la segunda entrada, de forma que si la primera es mayor que la segunda, la salida será un estado lógico alto (1), y en caso contrario la salida resultará un estado lógico bajo (0).

A.3.1.2.8 Modulador de anchura de impulso o PWM

Se trata de un mecanismo capaz de configurar ciclo de trabajo y el periodo de una señal. Consiste en el uso del contador y una comparación, el contador incrementa su valor conforme avanza el tiempo, y este valor es comparado con una señal o valor de referencia, con lo que se crea una señal como salida con un tiempo de estado lógico alto proporcional al valor de referencia e inversamente proporcional al tiempo de reinicio del contador, y un periodo acorde al tiempo que tarda el contador en reiniciarse.

A.3.1.2.9 Puertos de comunicación

A menudo los *microcontroladores* incorporan interfaces de comunicación, cuyo propósito es permitir la comunicación del *microcontrolador* con otros dispositivos de la misma o distinta naturaleza. Para llevar a cabo el intercambio de información destacan los siguientes recursos:

- *UART*, que posibilita la comunicación serie asíncrona a través de dos puertos, un transmisor (*TX* o *TXD*) y un receptor (*RX* o *RXD*), manteniendo compatibilidad con el protocolo *RS-232*C.
- *USART*, que facilita la comunicación serie síncrona y asíncrona, mediante los puertos *TX* y *RX*, usando protocolos estándar como el *RS-232C*.
- *PSP*, un puerto paralelo esclavo de 8 bits (datos) con señales de control (lectura *RD#*, escritura *WR#* y selección *CS#*) que permite la transmisión asíncrona.

- *USB*, un estándar serie industrial diferencial muy popular en los ordenadores personales, que utiliza dos líneas para la transmisión de datos (*D+* y *D-*).
- *Bus* I^2C, un *bus* de comunicaciones serie que usa dos líneas para transmitir información, una para la señal de reloj (*SCL*) y otra para los datos (*SDA*).
- *CAN*, un protocolo de comunicaciones diferencial muy empleado en automóviles en el que los datos se transmiten por *CAN+* y *CAN-*.

A.3.1.2.10 Interrupciones

Una interrupción consiste en un mecanismo por el que un evento puede detener la ejecución de un programa en cualquier instante para realizar un salto a una subrutina (*Interrupt Service Routine*). La subrutina, que se efectúan al aparecer una instrucción *CALL*, contiene el código necesario para reaccionar ante la interrupción o el evento y tras esto se retoma la ejecución del programa exactamente en el mismo punto donde había quedado suspendida.

Representan una herramienta muy importante para el control y la monitorización del *hardware* externo, ya que permiten la sincronización con los acontecimientos del sistema.

A.3.1.3 Desarrollo de software

Los *microcontroladores* son generalmente programados en un lenguaje de alto nivel, generalmente por lenguajes populares como *C, C++* o *Ada*, lo que facilita enormemente el desarrollo del programa del control y monitorización del *hardware*. Sin embargo, para conseguir esta comodidad es necesario un compilador cruzado, que transforma el lenguaje de alto nivel en el código entendible para el controlador, descargable en el *microcontrolador* para posteriormente ser ejecutado.

Existen numerosas metodologías para desarrollar el *software* que se encargan de aclarar y estructurar el proceso de desarrollo, pero todas estas metodologías presentan fases de diseño, implementación y depuración o testeo.

Tras la fase de diseño del programa es imprescindible descargar el código en el *microcontrolador* , bien mediante una interfaz de comunicaciones que requiere de un *software* especial (el programador) y un *hardware* dedicado (el adaptador de programación) o en su lugar a través de un *bootloader*, un programa que reside en la memoria del *microcontrolador* y se encarga de instalar los programas nuevos entrando en un modo de programación, descarga los programas desde el *PC* y los almacena en la memoria del *microcontrolador* .

El fichero descargado debe acabar en la memoria del controlador en formato binario, pero se suele emplear un fichero de formato extendido que muestra información acerca del tamaño del programa, su localización prevista y un *checksum* para asegurar la integridad; de manera que el programador o el *bootloader* traduce este formato en binario, por lo que la elección del formato corresponde al programador. Se suelen utilizar dos formatos, el formato de fichero

Hex (una serie de líneas en hexadecimal codificadas en ASCII) o el formato de fichero *S-Record* (un grupo de líneas o record).

A.3.1.4 Microcontroladores ARM

ARM es una arquitectura *RISC* de 32 bits desarrollada por *ARM*, que constituye un conjunto de instrucciones idóneo para aplicaciones de baja potencia, por lo que es la arquitectura dominante en el mercado de la electrónica integrada y móvil. Existen muchas empresas titulares de una licencia ARM para fabricar procesadores o *microcontroladores* con este tipo de arquitectura.

A.3.1.4.1 *Microcontrolador STM32F101V8*

STM32F101V8 es un *microcontrolador* de líneas de acceso de densidad media que incorpora el alto rendimiento de *ARM Cortex-M3*, un núcleo a 36MHz de frecuencia, memorias embebidas de alta velocidad (64 Kbytes de *Flash* y 10Kbytes de *SRAM*) y un amplio rango de periféricos y entradas/salidas conectados a dos *bus*es *APB*. Ofrece la integración de los interfaces de comunicación I^2C, *SPI* y *USART* y cuenta con 100 pines para gobernar sistemas con múltiples entradas/salidas. [36]

A.4 Cálculos

A.4.1 Cálculo de componentes

El sistema está constituido por numerosos elementos que no requieren ningún tipo de cálculo, son circuitos integrados acordes a la alimentación del sistema y su funcionalidad, pero los componentes que los acompañan (mayormente resistencias y condensadores), complementando o mejorando su funcionamiento, necesitan de cálculos que determinen sus características óptimas.

El tamaño de los componentes, concretamente de resistencias y condensadores, puede influir ligeramente en el diseño de la tarjeta de circuito impreso ofreciendo más dificultades (múltiples huellas, tamaños desmesurados, etc.), por lo que se emplea preferiblemente un encapsulado SMD pequeño, económico y adecuado a las características del sistema, el encapsulado 0603 (potencia típica en resistencia de100mW).

Todos los condensadores, excepto *C23* y *C24*, son utilizados para disminuir el rizado proveniente de la alimentación, tanto de *Vcc* (5V) como de *+3.3VDC*. Dado que cuanta mayor capacitancia menor rizado, es conveniente emplear condensadores de alto valor; sin embargo, los condensadores cerámicos (relativamente de menor capacitancia) son capaces de absorber cambios más bruscos, por lo que lo adecuado es usar condensadores cerámicos de alto valor (100nF) en las proximidades de cualquier circuito integrado alimentado a *Vcc* o *+3.3VDC*. Además, puesto que cada módulo de relés se sirve de un regulador para conseguir *+3.3VDC*, es apropiado añadir condensadores de aún más valor (10µF) y de cualquier tipo (tantalio), en paralelo con los condensadores cerámicos en el regulador, de manera que los cerámicos eliminen las variaciones de la tensión veloces y los condensadores de tantalio reduzcan los cambios de tensión elevados.

Si bien, ya ha sido determinado el valor, es preciso establecer el voltaje. El sistema maneja alimentaciones de 5V y 3.3V, es por ello que añadiendo un rango de seguridad, como mínimo el doble de la tensión de funcionamiento (10V) por ejemplo, cualquier condensador capaz de tolerar una tensión moderada es una elección adecuada.

La mayoría de las resistencias de los módulos de relés conforman *pull ups* o *pull downs*, que aseguran que en estados de salidas de control de alta impedancia o abiertos, toda posible fluctuación o tensión residual desaparezca gracias a la resistencia. El valor de estas resistencias se encuentran dentro de un rango determinado por un máximo y un mínimo, dependientes del tipo de *pull*:

- $R_{Pull-up,max} = \frac{V_S - V_{IH}}{I_{Pull-up}}$, siendo $I_{Pull-up}$ la corriente que atraviesa la resistencia de *pull up* cuando la salida de control está apagada y existe alimentación. $I_{Pull-up}$ se calcula mediante el uso de la ley de Kirchhoff y los valores de corriente *leakage* (I_{LKG}) de los *datasheets* de los circuitos integrados que intervienen en la línea. V_S es la alimentación en el caso de este sistema y V_{IH} es la tensión mínima especificada para ser reconocida como un estado lógico alto por el circuito integrado (entrada). [42]

- $R_{Pull-up,min} = \frac{V_S}{I_{Pull-up}}$, donde $I_{Pull-up}$ se calcula de la misma manera, pero en el momento en que la salida de control se encuentra activa, por lo que cambia su resultado. [42]
- $R_{Pull-down,max} = \frac{V_{IL}}{I_{Pull-down}}$, en el cual V_{IL} se encuentra establecido por el *datasheet* del circuito integrado (entrada) como el voltaje máximo a ser considerado un estado lógico bajo. $I_{Pull-up}$ se evalúa con la salida de control apagada, considerando la corriente del circuito integrado de salida como I_{OL} (corriente de salida de bajo nivel) y la corriente *leakage* del circuito integrado de entrada. [42]
- $R_{Pull-down,min} = \frac{V_{OH}}{I_{Pull-down}}$, en el que $I_{Pull-up}$ se supone en el instante en que la salida de control está activa (el datasheet presenta la corriente para un estado lógico alto, I_{OH}, y la corriente *leakage* correspondiente), y V_{OH} es el voltaje para una salida de estado lógico alto. [42]

Adicionalmente, el estándar *RS-485* establece una resistencia determinada por las fórmulas:

$R = \left(\frac{V_S}{V_{AB}} + 1\right) \cdot 27{,}8\Omega$ y V_{AB} = *Receiver input threshold* + noise

También determina una Resistencia mínima de 375Ω para los *pull ups* y *pull downs* que se emplean en las líneas.

El cálculo de componentes se ha organizado de acuerdo a la funcionalidad que desempeñan:

- *R1 (0R 0603)*. Representa un abierto con posibilidad de ser cortocircuitado por lo que no requiere cálculo alguno.
- *R2 (4K7 0603), R4 (4K7 0603), R3 (4K7 0603), R5 (4K7 0603), R9 (4K7 0603), R10 (4K7 0603), R11 (4K7 0603)* y *R12 (4K7 0603)*. Empleando las hojas de características de los circuitos integrados ADM1485ARZ se sustituye en las fórmulas:

  $V_{AB} \simeq 250$ ➔ $R = \left(\frac{V_S}{V_{AB}} + 1\right) \cdot 27{,}8\Omega \simeq 556\ \Omega$. [39]

  El resultado ofrecido supone el uso de una resistencia comercial de entre 510Ω y 680Ω, sin embargo si consideramos la posibilidad de que se conecten multitud de estos bloques en paralelo, tal y como sucede en el sistema, es necesario utilizar resistencias de más valor de modo que no circule tanta corriente a través de ellas. 4K7 supone un valor correcto para cumplir con estas directrices.

  En cuanto a la potencia que deben soportar, dado que $P = \frac{V^2}{R}$ y la tensión máxima a la que estarán sometidas es de 5V, P = 5.3mW, por lo que la potencia que soportan (ligada al encapsulado) no es determinante y se puede elegir el encapsulado *0603* (100mW), por ejemplo.
- *R6 (120R 1206), R7 (120R 1206), R17 (120R 1206)* y *R18 (120R 1206)*. Tal y como establece el estándar *RS-485,* estas resistencias tienen un valor de 120Ω. La potencia, $P = \frac{V^2}{R}$ = 208mw, implica el uso de un encapsulado mayor que la tolere, como mínimo el *1206* (250mW).
- *R8 (10K 0603)*. Consultando las hojas de características:

$R_{Pull-down,max} = \frac{V_{IL}}{I_{Pull-down}} = 25k\Omega$

$R_{Pull-down,min} = \frac{V_{OH}}{I_{Pull-down}} = 1{,}67k\Omega$

El valor apropiado se encuentra entre el máximo y el mínimo, por ejemplo 10kΩ.

La potencia, P = $\frac{V^2}{R}$ = 2.5mw, es prácticamente despreciable, por lo que no requiere adaptaciones en cuanto a encapsulado, el 0603 es más que suficiente.

- *D1 (VESD05A1-02V-GS08), D2 (VESD05A1-02V-GS08), D3 (VESD05A1-02V-GS08)* y *D4 (VESD05A1-02V-GS08)*. La elección de estos componentes de protección está unido a la tensión que suelen soportar las líneas, puesto que la máxima tensión diferencial ofrecida por los *transceivers* es de 5V, la mitad corresponde a la operación normal de cada línea independiente en la que se sitúa cada diodo. Un *reverse voltaje* del diodo (voltaje en el que el diodo conduce en la zona inversa), del doble del voltaje al que debe estar sometido (5V) es una opción adecuada para proteger la circuitería. [47]
- *L1 (MPZ2012S101A), L2 (MPZ2012S101A), L3 (MPZ2012S101A) y L4 (MPZ2012S101A).* Los *beads* ofrecen una fuerte resistencia ante frecuencias elevadas, mientras que en bajas frecuencias suponen un cortocircuito. El valor necesario para eliminar el ruido es un tanto subjetivo y debe ofrecer una resistencia baja ante la frecuencia de operación (115200 baudios). 100Ω ante una frecuencia de 100MHz, muy por encima de la velocidad de comunicación, es un valor aceptable para que a frecuencias más altas elimine eficazmente el ruido. [40]
- *D5 (APA2106CGCK), D6 (APA2106SYCK), D7 (APA2106SURCK), R23 (120R 0603), R24 (120R 0603) y R25 (120R 0603).* [13]
  Los diodos *LED* ofrecen una luminiscencia proporcional a la corriente, trabajando a una tensión determinada. De manera general, aunque luego se empleasen los modelos de la serie *APA2106*, se buscan diodos de tensión de trabajo a 2V (menor que la tensión ofrecida por las salidas del *microcontrolador*, 3.3V) y con una corriente de unos 10mA (por debajo del máximo de corriente de cada salida del *microcontrolador,* 25mA), lo que permite el cálculo de sus respectivas resistencias en serie:
  El voltaje que debe caer en la resistencia es V_R = 3.3V - V_{LED} = 1.3 V.
  La resistencia debe adaptar la corriente, I, que pasa por los *LEDs* a 10mA. Así:
  R = $\frac{V_R}{I}$ = 130Ω ➔ un buen valor comercial es 120Ω.
  La potencia, P = $\frac{V^2}{R}$ = 14mW, no representa ninguna restricción en cuanto a encapsulados empleados en el sistema, por lo que se utiliza por defecto el 0603 (100mW).
- *R26 (2K2 0603), R27 (2K2 0603), R28 (2K2 0603), R29 (2K2 0603)* y *R30 (2K2 0603)*. Son resistencias conectadas a un *switch (+3.3VDC*) encargadas de limitar la corriente que entra por cada entrada del *microcontrolador*. El máximo amperaje aceptable por cada entrada es de 25mA, por lo que a estos pines debería llegarles una tensión muy por debajo. Usando una resistencia de 2.2kΩ se garantiza una corriente muy inferior (sin tener en cuenta la resistencia interna de la entrada del *microcontrolador*, sería de unos 1.5mA).
  El valor de la potencia dista enormemente de la potencia ofrecida por el encapsulado *0603*, por lo que se emplea este.

- *R13 (2R7 0603), R13 (2R7 0603), R13 (2R7 0603)* y *R13 (2R7 0603)*. Son resistencias ocupadas de mejorar ligeramente la entereza de las comunicaciones, por lo que un valor bajo es suficiente para cumplir su cometido. La potencia que disipan estas resistencias es despreciable, ya que conforman divisores de tensión con los *pull ups* y *pull downs*, por lo que no es necesario calcularlas, se utiliza el encapsulado *0603*.
- *R19 (220R 0603) y R20 (1K 0603).* Desempeñan la labor de garantizar el estado lógico deseado. Dado que el estado de *BOOT0*, implica el cambio de modo *User Flash memory* (estado lógico 0) a *System memory* (estado lógico 1), es conveniente extremar las precauciones. *R19* constituye una resistencia de bajo valor que garantiza mejor integridad de señal y *R20* ejerce de *pull down,* aunque unidos constituyen un divisor de tensión que asegura el estado deseado. La potencia, teniendo en cuenta que establecen un divisor de tensión, $P_{R19} = \frac{V^2}{R} = \frac{(0.18 \cdot 3.3V)^2}{220\Omega}$ = 2mW y $P_{R20} = \frac{V^2}{R} = \frac{(0.82 \cdot 3.3V)^2}{1000\Omega}$ = 7mW, no excluye el encapsulado *0603*, por lo que se aplica este en ambos casos.
- *R21 (10K 0603)*. Es una resistencia a modo de *pull up*, pero como no se conocen los datos necesarios para los cálculos de los adaptadores de grabación y depuración y existen numeroso modelos de estos adaptadores, es conveniente emplear un valor convencional de *pull up* (10KΩ). La potencia P = $\frac{V^2}{R}$ = 2mW, lo que permite el uso del encapsulado *0603*.
- *R22 (1K 0603).* Según recomienda el fabricante *VBAT* debe ir conectado a una fuente de alimentación externa (*VDD*) en caso de que no se emplee su funcionalidad. Un cortocircuito a *+3.3VDC* podría provocar daños en el microcontrolador si se produjesen fluctuaciones elevadas, picos, etc. Para limitar la corriente se usa una resistencia de valor moderadamente alto y la potencia es a su vez despreciable frente a la capacidad de disipar la potencia del encapsulado *0603* de esta resistencia.
- C23 (12pF 0603) y C24 (12pF 0603). Estos condensadores forman parte del circuito oscilador del *microcontrolador*. Sus valores están determinados por la hoja de características del cristal (18pF), pero es conveniente usar una capacitancia ligeramente menor para asegurar que oscila a la velocidad esperada o aún más rápido, por lo que se emplea un valor de 12pF. La tensión que deben soportar es de unos 3.3V, es por ello que aplicando un margen de seguridad, cualquier condensador de más de 10V es apropiado.[1]
- *R31 (2K2 0603)… R82 (2K2 0603)*. Son resistencias *pull down* desde *R31* hasta *R82*, de modo que para calcular su valor:

  $R_{\mathrm{Pull-down,max}} = \frac{V_{IL}}{I_{\mathrm{Pull-down}}}$ = 18,8kΩ

  $R_{\mathrm{Pull-down,min}} = \frac{V_{OH}}{I_{\mathrm{Pull-down}}}$ = 1,12kΩ

  Una resistencia adecuada entre ambos valores es 2,2kΩ.

  La potencia, P = $\frac{V^2}{R}$ =11mW, muy inferior a la disipada por el encapsulado 0603, por lo que será su encapsulado.
- *R83 (1K 0603)* y *R87 (1K 0603)*. Estas resistencias identifican a los módulos de relés IND5003 e IND5002, respectivamente, *R83* en *CFG_SW2* y *R87* en *CFG_SW1*, mediante su conexión a *+3.3VDC*. Limitan la corriente que atraviesa las entradas del

microcontrolador, por lo que es idóneo un valor moderadamente alto. La potencia que disipan es muy inferior a la del encapsulado *0603*, así que se empleará este.

A.4.2 Cálculo de pistas

El cálculo de pistas es una labor que determina la anchura de las pistas para evitar sobrecalentamientos o daños en estas. Aunque puede suponer una tarea un tanto subjetiva en cuanto al establecimiento de los límites, existe un estándar general para el diseño de tarjetas de circuito impreso, el estándar *ANSI-IPC 2221* elaborado por *IPC* (*Association Connecting Electroning Industries*).

La corriente máxima (A), el incremento de temperatura (°C) y el grosor (altura) de la pista (Oz/Pie^2) basta para precisar el ancho de la pista:

$$\text{Ancho} = \frac{\text{Área}}{\text{L} \cdot 1{,}378}$$

Dónde L es el grosor de la pista y el Área = $\left(\frac{I}{k1 \cdot \Delta T^{k2}}\right)^{\frac{1}{k3}}$ en la cual:

I ➔ Corriente máxima

k1 ➔ Constante especificada por el estándar. 0,015 en el caso de ser una pista interna y 0,0647 cuando es una pista externa.

k2 ➔ Constante definida por el estándar. 0,5453 si se trata una pista interna y 0,4281 si resulta ser una pista externa.

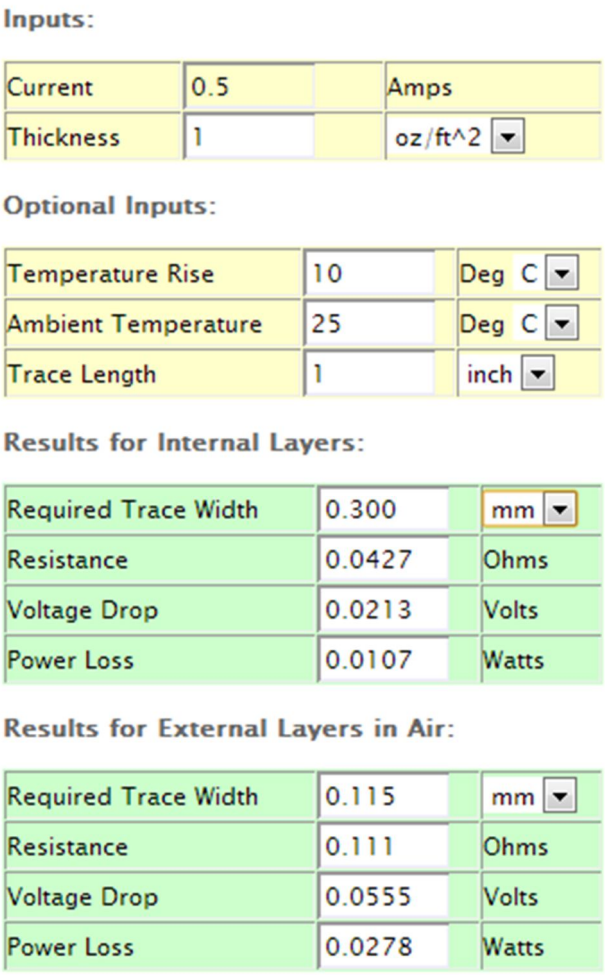
k3 ➔ Constante determinada por el estándar. 0,7349 para pistas internas y 0,6732 para pistas externas.

Si se aplica esta fórmula resulta:

$$\text{Ancho} = \frac{\left(\frac{I}{k1 \cdot \Delta T^{k2}}\right)^{\frac{1}{k3}}}{\text{L} \cdot 1{,}378}$$

Puede resultar un cometido laborioso, pero Internet ofrece numerosas páginas que calculan los resultados en base a estos términos y estándares, lo que redunda en un ahorro significativo de tiempo más aún cuando se debe elegir el grosor de la pista también. [4]

Las señales de control del sistema (*microcontrolador*, *drivers*, etc.) son del orden de miliamperios, es por ello que no hay que estudiar en profundidad estas anchuras. Por ejemplo los drivers pueden suministrar una corriente de hasta 500mA (350mA en el caso de este sistema):



Inputs:

Current	0.5	Amps
Thickness	1	oz/ft^2

Optional Inputs:

Temperature Rise	10	Deg C
Ambient Temperature	25	Deg C
Trace Length	1	inch

Results for Internal Layers:

Required Trace Width	0.300	mm
Resistance	0.0427	Ohms
Voltage Drop	0.0213	Volts
Power Loss	0.0107	Watts

Results for External Layers in Air:

Required Trace Width	0.115	mm
Resistance	0.111	Ohms
Voltage Drop	0.0555	Volts
Power Loss	0.0278	Watts

Figura 53. Cálculo de la anchura de las pistas de los *drivers* [4]

Dado que las pistas se implementan en las capas externas, debemos usar una anchura de 0.115mm para un grosor de 1 Oz/Pie^2 (35 micras).

Las señales realmente significantes en este estudio son las procedentes de los caminos de los relés, ya que en los módulos *IND5002* e *IND5003* permiten hasta 2A, en el módulo *IND2002A* no es necesario calcular estas pistas porque son reemplazadas por cableado. Así pues, con los términos conocidos:

Inputs:

Current	2	Amps
Thickness	1	oz/ft^2

Optional Inputs:

Temperature Rise	10	Deg C
Ambient Temperature	25	Deg C
Trace Length	1	inch

Results for Internal Layers:

Required Trace Width	2.03	mm
Resistance	0.00631	Ohms
Voltage Drop	0.0126	Volts
Power Loss	0.0252	Watts

Results for External Layers in Air:

Required Trace Width	0.781	mm
Resistance	0.0164	Ohms
Voltage Drop	0.0328	Volts
Power Loss	0.0656	Watts

Figura 54. Cálculo de la anchura de las pistas de los caminos de los relés con 1 $\frac{Oz}{Pie^2}$ [4]

Puesto que será necesario utilizar capas internas, habría que emplear pistas de una anchura de 2.03mm, lo que supone unas dimensiones abusivas teniendo en cuenta la cantidad de pistas a trazar. Es por ello que conviene el uso de más grosor de pista en los módulos *IND5002* e *IND5003:*

Inputs:

Current	2		Amps
Thickness	2		oz/ft^2

Optional Inputs:

Temperature Rise	10	Deg C
Ambient Temperature	25	Deg C
Trace Length	1	inch

Results for Internal Layers:

Required Trace Width	1.02	mm
Resistance	0.00631	Ohms
Voltage Drop	0.0126	Volts
Power Loss	0.0252	Watts

Results for External Layers in Air:

Required Trace Width	0.391	mm
Resistance	0.0164	Ohms
Voltage Drop	0.0328	Volts
Power Loss	0.0656	Watts

Figura 55. Cálculo de la anchura de las pistas de los caminos de los relés con 2 $\frac{Oz}{Pie^2}$ [4]

La anchura de las pistas se reduce a la mitad prácticamente (1.02mm), algo asumible en la tarjeta de circuito impreso. De esta manera, el grosor de las pistas de los módulos de relés se corresponde con 2 Oz/Pie^2 (70 micras) para los módulos *IND5002* e *IND5003* y 1 Oz/Pie^2 para el módulo *IND2002A.*

Si bien no es necesario calcular las pistas para el módulo de relés *IND2002A*, es imprescindible asegurar la sección del cableado. Los relés del módulo *IND2002A* podrían estar sometidos hasta por 16A y 250VAC, y dada su disposición en la *PCB* (al aire). Empleando el criterio por calentamiento y la norma *UNE 20460-5-523* (tablas de corrientes admisibles), resulta en una sección de 1,5 mm^2 (*AWG16*) para cualquier tipo de aislamiento y en cobre, apropiada para los pines del conector *GMCT41M*. [31]